Federal Aviation
Administration

U.S. Department of Transportation
Federal Aviation Administration

NAS Voice System

FAA NVS Addendum to EUROCAE ED-137A Part II

Override call requirements and specification

Federal Aviation Administration
800 Independence Avenue SW
Washington, DC 20591

Table of Contents

FIGURE INDEX

1.0 DEFINITIONS

The following definitions are used within this specification.

Call Transfer: a call feature that allows a user to redirect a G/G call that has either been answered or that is in the Common Answer queue at a given position to another position.

Ground-to-Ground (G/G): voice transmissions over IC, IP, or OVR using the voice switch, external networks, or using outside phone lines.

Hot Mic: a status of a microphone, headset or handset in which transmission is enabled without the selection of a push-to-talk button.

Indirect Access (IDA): a call mode wherein the call processing sequence required to establish a communication link or to select a control function is accomplished by entering multi-digit numbers on a remote keypad or a keypad on the position touch entry display. The keypad is activated by selecting the IDA mode.

Instantaneous Access (IA): a call mode wherein the entire call processing sequence required to establish circuit connectivity is accomplished as the result of a single touch action (G/G only).

Intercom (IC): a type of call that provides operator position intra-facility communications on a voice switch. Communications between controllers at the same facility.

Interphone (IP): a type of call that provides operator position inter-facility communications. Communications between controllers at different facilities.

Latching - a function that either is or emulates a pushbutton that locks in the down position upon a first touch, and requires a second touch to release the locked condition. The desired activation is in effect for the time the button is in the locked position.

Non-Latching - a feature which either is or emulates a pushbutton that requires an operator to provide continuous touch action to maintain the desired pushbutton activation. The activation is terminated by the release of touch action on the pushbutton.

Override (OVR): a call feature whereby a call being placed results in connection to the called party, even if the called position has an active call in progress.

Override Call Chaining: a situation where party A overrides party B who is overriding party C, etc.

Party - a person or group of persons who participate in a G/G call or conference; the person under consideration.

Party (A-party, B-party and C-party): The users involved, sequentially, in a telephone call as follows:
A-party: the user who initiates a call (the calling party);
B-party: the user who first receives the call (the called party);
C-party: the third user involved in the call;
D-party: the fourth user involved in the call;

Pre-empt: The disconnection and subsequent reuse of part or all of an established connection of lower priority origins by a higher priority source.

2.0 FAA OVERRIDE CALL REQUIREMENTS

In case of any resulting conflict in specification between the EUROCAE ED-137A Part II document and this FAA addendum for OVR Call requirements and specification, the contents of this document **SHALL** prevail.

2.1 FAA Override Calls (OVR)

The OVR call feature can be employed where there is an operational requirement for this and is a means of automatically ‘establishing’ a two-way telephone communication between two Users.

The OVR call allows the calling user to listen-in to the sum of the following audio at the called operator position once the call is established:

1. the combined audio from all active Ground-Ground and Air-Ground communications in progress at the called operator position plus audio from the called operator position microphone (i.e. Called user Hot-mic)
2. the combined audio from all active Ground-Ground and Air-Ground communications currently being received by the called operator position as a result of its own established outgoing override call and incoming override calls;

The OVR call allows the called user to hear the sum of the following audio at the calling operator position when the OVR call is connected.

1. The combined audio from all active Ground-Ground communications being received by the calling operator position as a result of one or multiple established incoming override calls plus the calling operator position microphone (i.e. Calling user Hot-mic);

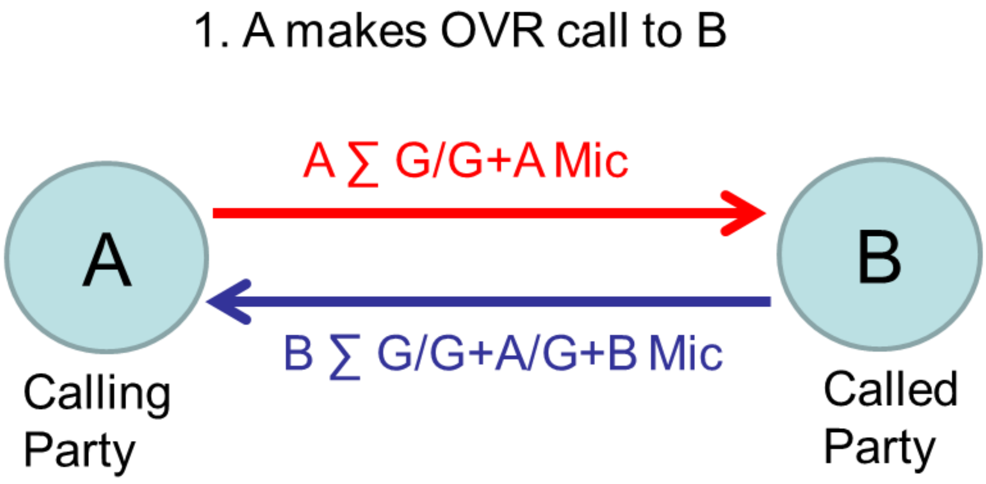


Figure 1 - Audio flow example for OVR call from A to B

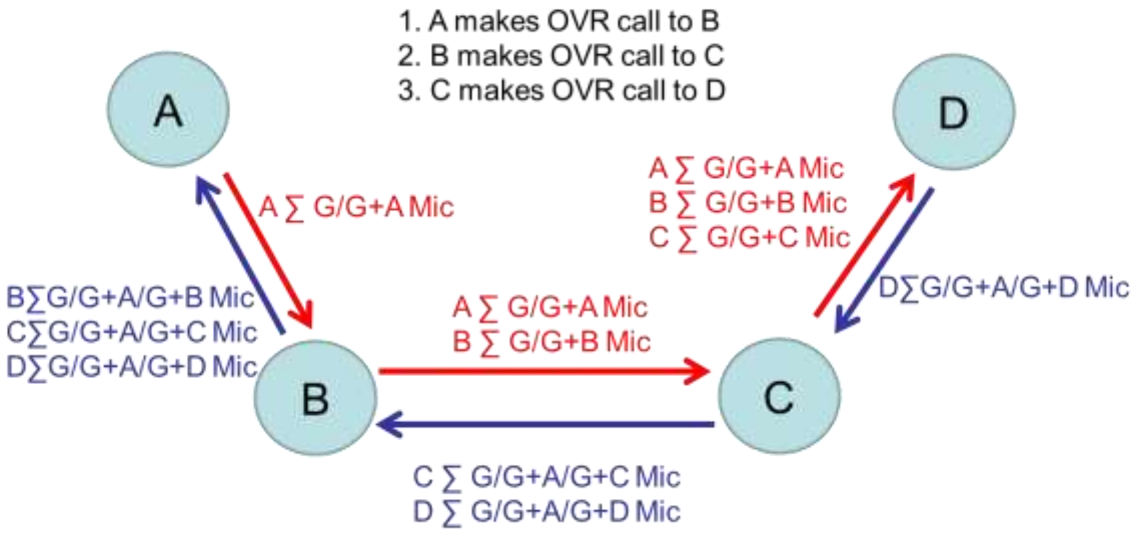


Figure 2 - Audio flow example for OVR calls from A to B, B to C and C to D.

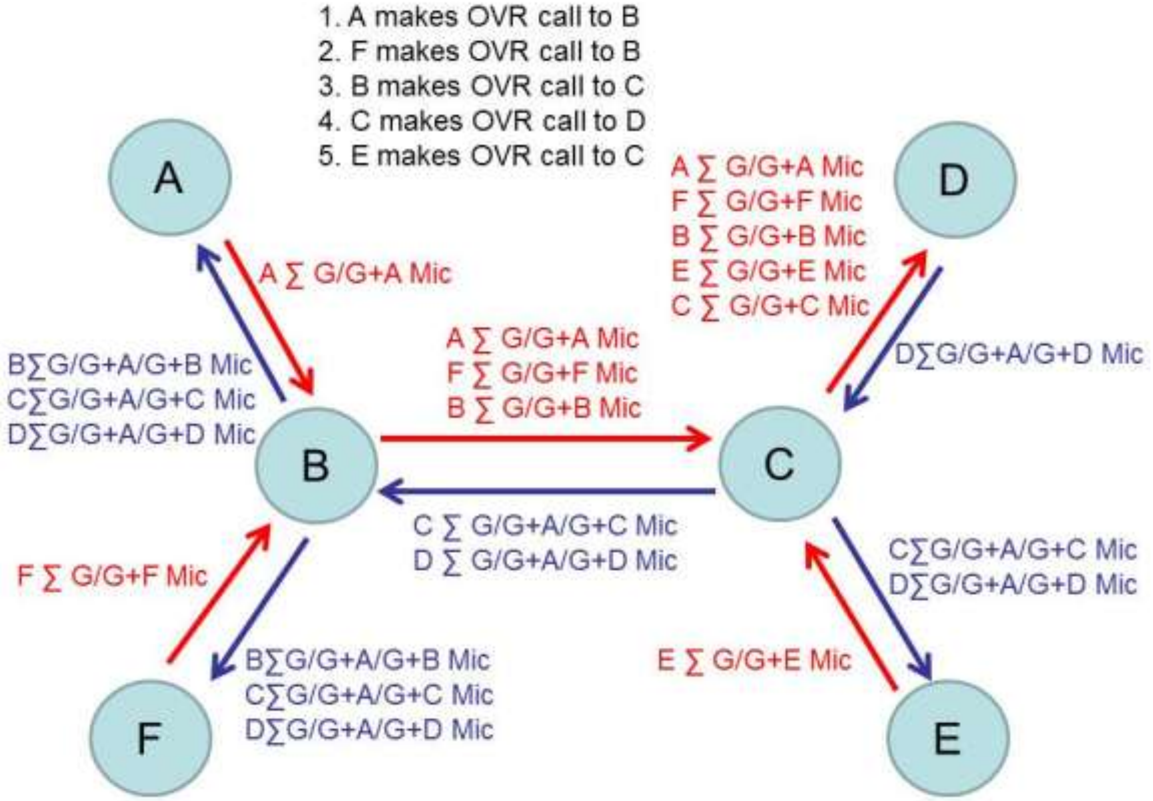


Figure 3 -Audio flow example for OVR calls from A to B, F to B, B to C, C to D & E to C.

1 **[REQ TEL FUNCTIONAL] OVERRIDE CALLS**

The Override call is one of the principal call types used by the Federal Aviation Administration (FAA). It is used in the same manner in both ACCs and Approach environments.

An OVR call can be made between two parties at the same ATC site (intra-facility call) or between two parties at different ATC sites (inter-facility call).

The requirements in this section have the scope of defining OVR call operation between multi-vendor systems located at different ATC sites (i.e. inter-facility call) in order that the OVR call specification (provided in section 3) can define the protocol to be applied in order to achieve interoperability between multi-vendor systems.

2.1.1 OVR call description

1. An OVR call **SHALL** be initiated by the ‘A’-party either through operation of a single touch to a latching or non-latching Instantaneous Access (IA) key (on the touch panel at the operator position) or through entry of the OVR call access code followed by Called party address via the Indirect Access (IDA) keypad at the operator position.

2. In the case of an Instantaneous Access (IA) key, the 'B'-party address **SHALL** be assigned and configured (by means of system management), in the 'A'-party VCS and is thus uniquely associated with a particular key labeled with the ‘B’-party’s identity.

3. The 'A'-party **SHALL** be able to make only one outgoing OVR call at a time.

4. The 'B' party **SHALL** be able to receive one or more OVR calls up to a maximum limit defined by system requirements.

5. Although a 'B'-party is able to receive multiple incoming OVR calls, it **SHALL NOT** be prevented from being a calling party. It is allowed to make a single outgoing OVR call to another party (i.e. ‘C’-party).

6. When an Instantaneous Access key is used for making an OVR call, dial tone and out-going signaling tones **SHALL NOT** be given to the 'A'-party.

7. Ringback tone **SHALL NOT** be given to the 'A'-party.

8. Terminal Out-of-service tone **SHALL** be given to the ‘A’-party should the OVR call fail for any reason.

9. The 'A'-party identity **SHALL** be indicated to the 'B'-party either by association with a key assigned on the 'B'-party touch panel or by means of a dynamic

display. Any visual (and/or audible) alerts associated with an incoming OVR call **SHALL** be distinctive from other types of call.

10. An audible alert **SHALL** be generated at the 'B'-party on arrival of an incoming OVR call.

11. The 'B'-party **SHALL** automatically accept the incoming OVR call without any intervention required by the User; this **SHALL** occur regardless of the 'B'-party being engaged in any other type of ground-ground call or air-ground call. When the OVR call is automatically answered by the 'B'-party, a two-way speech communication between the 'A'-party and the 'B'-party **SHALL** be established.

12. The establishment of an OVR call from the 'A' party to the ‘B’-party **SHALL** result in the 'A' party sending a single audio stream towards the ‘B’-party that contains the sum of audio resulting from all Ground-Ground calls in progress at the 'A' party, currently being forwarded to the 'A' party by other parties. This stream **SHALL** therefore include G/G audio from all established incoming OVR calls at the 'A' party as well as all speech (Hot-mic) at the 'A' party microphone.

13. The establishment of an OVR call from the 'A' party to the ‘B’-party **SHALL** result in the 'B' party sending a single audio stream towards the 'A' party for monitoring purposes, that contains the sum of all Ground-Ground audio plus all Air-Ground audio plus Microphone speech in progress at the 'B' party (Hot-Mic) as well as the monitored audio stream that the 'B' party may receive as a result of its own outgoing OVR call to a third party (i.e. ‘C’ party) and incoming calls from third parties.

    Note 1.
    When an Override call is established, a calling user forwards to the called user the sum of audio from existing Ground-Ground communications only plus the calling user microphone, while the called user returns to the calling user the sum of all existing Ground-Ground and Air-Ground communications plus called user microphone.

14. If the 'B'-party’s A/G selector switch is set to Headset, the 'A' party making an OVR call to the 'B' party **SHALL** directly hear pilot speech arriving at the earpiece of the 'B' party’s headset.

15. Speech from the 'A'-party towards the 'B'-party **SHALL NOT** be connected to the air-ground uplink of the 'B'-party. The pilot **SHALL NOT** therefore hear any speech from the 'A'-party.

16. Speech from the 'A'-party **SHALL** be directed to the 'B'-party’s headset in the case that the 'B'-party’s OVR call selector switch is set to headset (this is the default setting). Speech from the 'A'-party **SHALL** however be directed to the

'B'-party's loudspeaker in the case that the 'B'-party's OVR call selector switch is set to loudspeaker.

17. It **SHALL NOT** be possible for the 'B'-party to disabled the Monitoring of any speech to the 'A'-party directly from its position.

18. In order for the 'B'-party to respond to the 'A'-party no key activation associated with the incoming OVR call is necessary, because a two-way voice channel has already been established. The return monitored speech path **SHALL** therefore always be enabled from the 'B'-party towards the 'A'-party while the OVR call is active.

19. All speech spoken at the 'B'-party's headset microphone **SHALL** be heard by the 'A'-party even when there are no other active telephone or radio calls in progress at the 'B'-party side.

    Note 2.
    FAA Operational procedures require that when an OVR call is connected, the 'A'-party verbally announces the purpose of its call to the 'B'-party. The 'B'-party has the responsibility to prioritize the handling of multiple incoming OVR calls, verbally identifying a specific 'A'-party prior to giving its permission to continue speaking. If the 'A'-party requires to urgently talk with the 'B'-party, the 'A'-party is allowed to verbally interrupt any ongoing conversation in the conference, but has to follow a pre-defined verbal protocol by previously announcing the words "Break for Control".

20. A chain of OVR calls would occur if A calls B, B calls C and C calls D etc. In case A makes OVR call to B, B makes OVR call to C, then:

    - A will hear G/G and A/G audio from B and C (as summed audio on monitoring path);
    - B will hear G/G and A/G audio from C (as summed audio on monitoring path) and will hear G/G audio from A (as summed audio on forward path);
    - C will hear G/G audio from A and B (as summed audio on forward path).

    All users **SHALL** therefore hear all other users.

21. For the case that establishment of an OVR call would form a loop (i.e. A calls B, B calls C and then C calls A), a loop closure detection algorithm **SHALL** be implemented in the protocol in order to prevent operators from hearing acoustic feedback. Once an OVR call is made that will cause a loop to occur, then actions **SHALL** be taken in order to prevent acoustic feedback caused by audio circulating continuously around a loop of audio paths formed by establishment of a loop of OVR calls.

This will imply blocking summed audio streams at the point where a loop detection has occurred from being passed on to both the successive and previous VCS systems.

22. Once an OVR call is established it **SHALL** be protected from call pre-emption.

23. Once an OVR call is established it **SHALL** be forbidden to execute a call transfer of the OVR call.

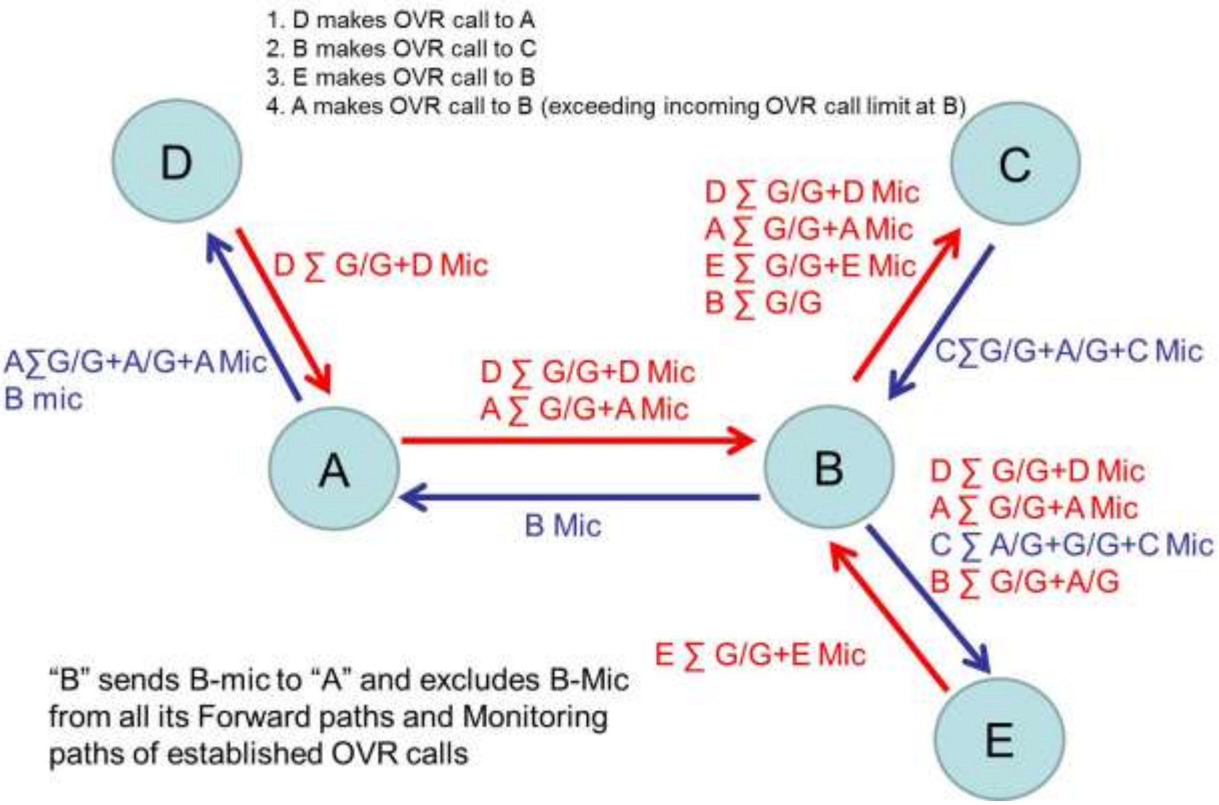
24. Once an OVR call is established it **SHALL** be forbidden to place the OVR call on hold.

25. Clearing the OVR call **SHALL** occur from either the 'A'-party side or from the ‘B’-party side. It **SHALL** be possible for the 'B'-party to clear an established OVR call made from the 'A'-party side.

26. An incoming OVR call that exceeds the 'B'-party limit for connected OVR calls **SHALL NOT** be given busy tone, but **SHALL** cause the establishment a of two party call between the ‘A’ and ‘B’ parties while the original OVR conference is automatically put on Hold by the ‘B’-party for the duration of the call. A warning message to this affect shall be forwarded to a Supervisor position.

    Note 3.
    The ‘B’-party is aware that the ‘B’-party limit for connected OVR calls has been exceeded. The ‘B’-party will only send the ‘B’-party microphone audio to the ‘A’-party over the monitoring channel, excluding any G/G and A/G audio. The ‘B’-party will send the sum of its G/G +A/G audio on the monitoring channel to all other established Incoming OVR calls (excluding the ‘B’-party microphone from the sum).
    The ‘A’-party will forward G/G audio + ‘A’-party microphone audio to the ‘B’-party. If the ‘B’-party had previously established an outgoing OVR call with a ‘C’-party, the ‘B’ party will forward the sum of all G/G audio + Microphone audio received from all of its established incoming OVR calls to the ‘C’-party (excluding the ‘B’-party microphone from the sum).

B-party maximum OVR call limit exceeded example



1. D makes OVR call to A
2. B makes OVR call to C
3. E makes OVR call to B
4. A makes OVR call to B (exceeding incoming OVR call limit at B)

“B” sends B-mic to “A” and excludes B-Mic from all its Forward paths and Monitoring paths of established OVR calls

2.1.2 Performance Criteria

1 [REQ TEL PERFORMANCE] OVERRIDE CALL PERFORMANCE CRITERIA

(a) Override call is designed to meet the requirements of ICAO 9804 Instantaneous Controller-Controller Voice Communication (ICCVC) which stipulates that two-way direct communication **SHALL** be established between non-physically adjacent controllers within 1 second or less in 99% of the time.

(b) The above interval is the delay between the 'A'-party initiating the call and the 'A'-party to 'B'-party two-way speech path being established. A more stringent value of 200ms maximum is imposed by the FAA as an NVS requirement.

(c) OVR call **SHALL** be clearly distinguishable from a Direct Access (DA) call, IDA (Indirect Access Call) and the European Instantaneous Access (IA) call in any signaling protocol.

(d) The minimum limit for incoming simultaneous OVR calls to a single operator position **SHALL** be set to 9.

(e) The maximum limit for incoming simultaneous OVR calls to a single operator position **SHALL** be set to a system configurable maximum limit. The FAA/ANSPs **SHALL** have the possibility to reduce this maximum limit.

(f) Any attempted Override call beyond that of the system configurable maximum limit **SHALL** result in the establishment of a two-party call between the calling and called operator positions while the original OVR conference is automatically put on Hold. A warning message to this affect **SHALL** be forwarded to a Supervisor position.

Note 4.
The System configurable maximum limit is set to 9.

3.0 FAA OVERRIDE CALL SPECIFICATION

3.1 Override Call

An "Override call" is a call that enables two-way audio communication between calling party and the called party, without the called party having to perform any acceptance action at its operator position.

An OVR call can be made between two parties at the same ATC site (intra-facility call) or between two parties at different ATC sites (inter-facility call). This specification has the scope of defining the protocol to be applied in order to achieve OVR call interoperability between multi-vendor systems (inter-facility call) according to the requirements defined in the previous section. It does not have the scope of defining the protocol to be applied to intra-facility OVR calls.

In order to make an OVR call, the calling party will either activate the associated IA key on the touch panel at its terminal or enter the OVR call access code followed by the Called party address via the Indirect Access (IDA) keypad at the operator position. The called party is informed of OVR call arrival through a change in state of the corresponding IA key on the touch panel and a visual OVR call indication at its operator position.

On establishment of an incoming OVR call from the calling party, the VCS system places the calling party in a conference type mode thereby allowing the calling party to hear audio from all active ground telephone calls and air-ground transmissions and receptions emanating from the called operator position, plus the audio from the called party microphone.

Likewise on arrival of an incoming OVR call from the calling party, the calling party shall send the called party the summed audio from all active ground telephone calls received at the calling operator position plus the audio from the calling party microphone.

Audio from the called party headset microphone is always being sent to the calling party even in the case that there are no other active ground telephone or radio calls in progress at the called party side. It is not necessary for the called party to activate any key (as in the case of a European IA call). Likewise audio from the calling party headset microphone is always being sent to the called party even in the case that there are no other active ground telephone calls in progress at the calling party side.

The OVR call remains connected until either the calling party or the called party clears the call by deselecting the associated IA key or by selecting the Call Release key or by answering another G/G call.

Note 5.

The Override call is similar to the Instantaneous Access call and can be implemented by using the Instantaneous Access supplementary service as defined by ED-137A Part 2 Telephone. The called party to an incoming OVR call will however always accept a two-way session by using the SDP "send-receive mode" attribute set to sendrecv and the SDP *"service" attribute set to "a=service: duplex". The OVR call also introduces new SDP "a=sid", "a=rid" and "a=service" attributes.*

3.1.1 Message Body

SIP message bodies containing a description of the session, time and media **SHALL** be encoded in the Session Description Protocol (SDP) (RFC 4566). The SDP types and parameters indicated in **Error! Reference source not found. SHALL** be supported. SDP parameters and values **SHALL** be interpreted as being case sensitive.

Table 4 – Supported SDP Types and Parameters

Description	SDP Types	SDP Parameters	Values
Session	Protocol version ("v=")	<SDP version number>	0
	Origin ("o=")	<username>	(Application dependent)
		<sess-id>	(Application dependent)
		<sess-version>	(Application dependent)
		<nettype>	IN
		<addrtype>	IP4 (in the interim) IP6
		<unicast-address>	(Application dependent)
	Session name ("s=")	<session name>	(Application dependent)
	Connection data ("c=")	<nettype>	IN
		<addrtype>	IP4 (in the interim) IP6
		<connection-address>	(Application dependent)
Time	Timing ("t=")	<start-time>	(Application dependent)
		<stop-time>	(Application dependent)
Media	Media descriptions ("m=")	<media>	audio
		<port>	(Application dependent)
		<proto>	RTP/AVP
		<fmt>	0 (for PCM-μ) 8 (for PCM-A) 15 (for G.728) 18 (for G.729)
	Attributes ("a=")	<send-receive mode>	recvonly sendrecv sendonly inactive
		rtpmap:<payload type> <encoding name>/<clock rate> **(optional)**	PCMU/8000 (for PCM-μ) PCMA/8000 (for PCM-A) G728/8000 (for G.728) G729/8000 (for G.729)
		a=sid:<unique node id> **(optional)**	(Application dependent)
		a=rid:<unique node id> **(optional)**	(Application dependent)
		a=service:<type> **(optional)**	simplex duplex limited

3.1.2 Establishment Conditions

The IA Supplementary Service (SS-IA) must be available in the VCSs involved.

An ‘A’-party can make an OVR call to a ‘B’-party operator position without a pre-configured IA key present. The OVR call will be routed to a key in the Indirect Access call queue with a visual indication given for OVR call presence.

The ‘B’-party may be free or involved in another active call (e.g. OVR, Radio, DA/IDA routine or DA/IDA priority call); the actual status of the ‘B’-party is irrelevant.

3.1.3 Audio

Once the OVR call is established at the 'B'-party, because monitoring is always enabled, the 'A'-party **SHALL** hear a combination of:

- sent and/or received audio from any active incoming OVR telephone calls at the 'B'-party (in the case that the 'B'-party’s OVR selector switch is set to headset); and
- sent and/or received audio from any active incoming DA/IDA telephone calls at the 'B'-party; and
- sent and/or received audio from any active outgoing OVR telephone call at the 'B' party to a ‘C’-party. The received audio may include combined G/G and A/G audio directed to the ‘B’ party from the ‘C’-party.
- transmitted audio from any active radio call at the 'B'-party’s microphone; and
- received audio from any active aircraft call at the 'B'-party: and
- any other audio signal suitably injected in the audio path to the 'A'-party.

Once an OVR call from the ‘A’-party is established, the ‘B’-party **SHALL** hear a combination of:

- sent and/or received audio from any active incoming OVR telephone calls at the 'A' party (in the case that the 'A' party’s OVR selector switch is set to headset) ; and
- sent and/or received audio from any active incoming DA/IDA telephone calls at the 'A' party; and
- any other audio signal suitably injected in the audio path to the 'B' party.

In the case that the ‘A’-party has one or several incoming OVR calls established, the ‘B’-party **SHALL** also hear the summed audio from all of these parties with calls established to the ‘A’-party.

The ‘B’-party **SHALL NOT** hear:

- transmitted audio from any active radio call at the 'A' party’s microphone; and
- received audio from any active aircraft call at the 'A' party;

In the case that the ‘B’-party has made its own OVR call to a ‘C’-party while there are already established incoming OVR calls at the ‘B’-party position, the ‘C’-party **SHALL** hear audio coming from any of the ‘B’-party’s incoming OVR calls.

The ‘B’-party **SHALL NOT** transmit audio from any of the established OVR calls on its air-ground uplink.

In order to prevent either echoing or audio feedback, VCS systems **SHALL** implement an audio loop detection method in order to avoid possible problems derived from the same audio coming to a party’s operator position through two different RTP paths. This audio loop detection method shall prevent a loop from occurring in a chain of OVR calls (see 3.1.7).

3.1.4 Timing Constraints

The SS-IA **SHALL** meet the ICAO 9804 defined requirements of the Instantaneous Controller-Controller Voice Communication (ICCVC) which stipulates that communication be established between non-physically adjacent controllers within 1 second or less in 99% of the time.

The above interval is the delay between the 'A'-party initiating the call and the 'A'-party to 'B'-party two-way speech path being established. A more stringent value of 200ms maximum is imposed by the FAA as an NVS requirement.

3.1.5 OVR Call Signaling Procedures

The OVR call **SHALL** be handled as one session (A ↔ B) initiated by its creator. The OVR call session from ‘A’-party to ‘B’-party **SHALL** be released (cleared) either by the ‘A’-party or by the ‘B’-party.

A VoIP VCS implementing SIP SS-IA **SHALL** support the offer/answer model specified in RFC 3264.

3.1.5.1 Actions at the Calling-party User Agent (UA)

To make an OVR call, the ‘A’-party User Agent **SHALL** send an INVITE request, including a Priority header field with value “urgent” and a Subject header field with value “IA call”; it **SHALL** start timer T0.

The INVITE’s Session Description Protocol (SDP) message body **SHALL** contain an “a=sendrecv” SDP offer attribute and the SDP “a=service” attribute set to “a=service:duplex”. In addition, the offer shall contain the SDP “a=sid” attribute set to “a=sid:ID0 ID1 ID … IDn” containing all established OVR call members. The identifier from call participants of incoming OVR calls **SHALL** be unique over space and time and listed as call participants source ID separated by a single SPACE character, forming the OVR tail.

Note 6.
Each node shall provide an OVR tail and chain to its adjacent node. The tail comprises all nodes that form a branch terminating at the overridden node (‘B’-party). The chain comprises all nodes that form a branch originating from an overriding node (‘A’-party).
In general, a tail is being forwarded via INVITE and INFO messages containing the sending node identifier (ID0) and the sum of all tails connecting to this node (ID1 ID2 .. IDn) as a list within the SDP attribute “a=sid:ID0 ID1 ID2 .. IDn”. The chain is being forwarded via 200 OK (as final response to the INVITE request) and INFO messages containing the sending node identifier (ID0) and the chain connecting to this node (ID1 ID2 .. IDn).

The SDP parameter “a=service:duplex” and “a=sid:ID0 ID1 ID2 …IDn” are mandatory in INVITE, 200OK and INFO SIP messages of an OVR call.

If the call fails for reasons other than those covered below, the ‘A’-party UA **SHALL** stop timer T0 and indicate “OVR call failure” to the ‘A’-party.

While awaiting OVR call confirmation,

a) if the ‘A’-party releases the OVR call, the ‘A’-party UA **SHALL** send a CANCEL request and stop timer T0;

b) on receipt of a 200OK final response containing “a=sendrecv”, “a=service: duplex” and “a=sid:ID0 ID1 ID2 …IDn” answer SDP attributes, the ‘A’-party UA **SHALL** send an ACK request and establish both-way RTP media, enable ‘A’-party’s Tx and Rx paths, stop timer T0 and enter state IA-Tx-Active; “IA-Tx: active, IA-Rx:active” **MAY** be indicated on the appropriate IA key on the touch panel of the ‘A’-party’s operator position;

Note 7.
At this point, a session has been created in which the voice path from the ‘B’-party to ‘A’-party is used for monitoring audio of Ground and Radio calls in progress at the ‘B’-party’s operator position and for monitoring any audio at the ‘B’-party’s microphone even when there are no other Ground and Radio calls in progress at the ‘B’-party’s operator position.
The voice path from the ‘A’-party to ‘B’-party is used for sending summed audio of Ground calls in progress at the ‘A’-party’s operator position and for sending any audio at the ‘A’-party’s microphone even when there are no other Ground and Radio calls in progress at the ‘A’-party’s operator position.

c) on receipt of a 4xx-6xx final response, the ‘A’-party UA **SHALL** indicate “OVR call failure” to the ‘A’-party and stop timer T0;

d) on expiry of timer T0, the ‘A’-party UA **SHALL** indicate “OVR call failure” to the ‘A’-party and **SHALL** send a CANCEL request.

e) on receipt of a 200OK final response containing “a=sendrecv”, “a=service:limited” and “a=sid:ID0 ID1 ID2 .. IDn” answer SDP attributes, the ‘A’-party UA **SHALL** send an ACK request and establish both-way RTP media, enable ‘A’-party’s Tx and Rx paths, stop timer T0 and enter state IA-Tx-Active; “IA-Tx: active, IA-Rx:active” **MAY** be indicated on the appropriate IA key on the IA panel of the ‘A’-party’s CWP. In this case the OVR call limit has been exceeded at the ‘B’-party, implying that only a simple 2-party call is established. The ‘A’-party **SHALL** take no action with respect to processing the sid attribute. The 2-party call remains until cleared. .

Once the OVR call is established,

a) the ‘A’-party **SHALL** send the Sum of ‘A’-party’s G/G audio plus the ‘A’-party microphone audio (A∑G/G +A mic) to the ‘B’-party. If the ‘A’-party also has incoming OVR calls connected, it **SHALL** also add the G/G audio plus the microphone audio from each of the calling parties to the audio sum being sent to the ‘B’-party.

b) the ‘A’-party **SHALL** issue a SIP INFO message (text/plain) containing its updated chain to each established incoming OVR call (if any exist), in order to provide calling parties with updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the monitoring path to each calling party should reflect connections present in the chain originating from the ‘A’-party.

c) should the ‘A’-party receive a SIP INFO message (text/plain) defining an update to the local chain of its ‘B’-party, it **SHALL** issue a SIP INFO message (text/plain) containing its own updated chain to each established incoming OVR call (if any exist) in order to provide calling parties updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the monitoring path to each calling party should reflect connections present in the chain originating from the ‘A’-party.

d) should the ‘A’-party receive a SIP INFO message (text/plain) defining an update to the local tail of its calling parties, it **SHALL** issue a SIP INFO message (text/plain) containing its own updated tail to the established outgoing OVR call in order to provide the called party updated information about its current audio connections (see 3.1.7). The summed audio stream being sent to the called party should reflect connections present in the tail terminating at ‘A’-party.

e) if the ‘A’-party clears its outgoing OVR call, the ‘A’-party UA **SHALL** send a BYE request in order to clear the session. This **SHALL** be acknowledged by the ‘B’-party with a 200OK final response.

f) if the ‘B’-party clears its incoming OVR call, the ‘B’-party UA **SHALL** send a BYE request in order to clear the session. This **SHALL** be acknowledged by the ‘A’-party with a 200OK final response.

If a two-party call is established because the ‘B’ party OVR call limit has been exceeded,

a) the ‘A’-party **SHALL** send the ‘A’-party’s microphone audio (A mic) to the ‘B’-party. If the ‘A’-party also has incoming OVR calls connected, it **SHALL** add the G/G audio plus the microphone audio from each of the calling parties to the audio sum being sent to the ‘B’-party.
b) the ‘A’-party **SHALL** include the received ‘B’-party microphone audio (B mic)

with the summed audio stream being sent on the monitoring path to each of its calling parties.

Once the OVR call has been cleared by either the ‘A’-party or the ‘B’-party,

a) the ‘A’-party **SHALL** issue a SIP INFO message (text/plain) containing its updated chain to each of its established incoming OVR calls (if any exist), in order to provide calling parties updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the monitoring path to each calling party should reflect connections present in the chain originating from the ‘A’-party.

3.1.5.2 Actions at a Proxy

No special actions are required in support of SIP SS-IA.

3.1.5.3 Actions at the Called-party User Agent (UA)

Upon receipt of an INVITE (Priority=“urgent”, Subject=“IA call”) request with an SDP message body containing the parameters “a=sendrecv”, “a=service:duplex” and “a=sid:ID0 ID1 ID2 …IDn” as offer attributes, the ‘B’-party UA **SHALL** recognize this as an OVR call and check if there is an application dependent cause to reject the OVR call;

1. Should there be a cause for rejection, the ‘B’-party UA **SHALL** send the corresponding 4xx-6xx response; “Incoming call attempt” **MAY** be indicated on the appropriate IA key on the touch panel of the ‘B’-party’s operator position;

2. otherwise, the ‘B’-party UA **SHALL** immediately send a 100 Trying followed by a 200 OK response containing the “a=sendrecv”, “a=service:duplex” and “a=sid:ID0 ID1 ID2 …IDn” containing all established OVR call members (identifier from call participants of the outgoing OVR call unique over space and time) listed as call participants source ID separated by a single SPACE character, forming the OVR chain (if one exists).“IA-Tx: active, IA-Rx: active” **MAY** be indicated on the appropriate IA key on the touch panel of the ‘B’-party’s operator position.

3. Should the arrival of the OVR call cause the OVR call limit configured at the ‘B’-party to be exceeded, the ‘B’-party **SHALL** allow only a 2-party call to be established. In this case the ‘B’-party UA **SHALL** immediately send a 100 Trying followed by 200 OK response containing the “a=sendrecv”, “a=service:limited” and “a=sid:ID0 ID1 ID2 .. IDn” containing all established OVR call members.

Once the OVR call is established,

a) the ‘B’-party **SHALL** send the Sum of ‘B’-party’s G/G plus A/G audio plus the ‘B’-party microphone audio (B∑G/G + A/G + B mic) on the monitoring path to the ‘A’-party and also on the monitoring path to the calling parties of any other of its established incoming OVR calls (if they exist). If the ‘B’-party also has its own outgoing OVR call connected to a ‘C’-party, it **SHALL** also add the G/G +A/G audio plus the ‘C’-party microphone audio received from the ‘C’-party to the Audio sum being sent on the monitoring path towards the ‘A’-party and also on the monitoring path to the calling parties of any other of its established incoming OVR calls (if they exist).

b) the ‘B’-party **SHALL** also sum the received ‘A’-party G/G audio plus the ‘A’-party microphone (A∑G/G +A mic) with its own G/G audio plus the ‘B’-party microphone (B∑G/G +B mic) and also add the G/G audio plus microphone audio from the calling parties of any of its incoming OVR calls. This combined audio stream **SHALL** be sent on the forward path of its established outgoing OVR call towards the ‘C’- party.

c) the ‘B’-party **SHALL** issue a SIP INFO message (text/plain) containing its updated tail (including all tails terminating at the ‘B’-party) to its established outgoing OVR call with the ‘C’-party (if it exists), in order to provide the ‘C’-party with updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the forward path to the ‘C’-party should reflect connections present in the tail terminating at the ‘B’-party.

d) should the ‘B’-party receive a SIP INFO message (text/plain) defining an update to the local chain of its ‘C’-party, it **SHALL** issue a SIP INFO message (text/plain) containing its own updated chain to each established incoming OVR call (if any exist) in order to provide calling parties updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the monitoring path to each calling party should reflect connections present in the chain originating from the ‘B’-party.

e) should the ‘B’-party receive a SIP INFO message (text/plain) defining an update to the local tail of its calling parties, it **SHALL** issue a SIP INFO message (text/plain) containing its own updated tail (including all tails terminating at ‘B’-party) to the established outgoing OVR call in order to provide the ‘C’-party updated information about its current audio connections (see 4.4.7). The summed audio stream being sent to the ‘C’ party should reflect connections present in the tail terminating at the ‘B’-party.

f) on receipt of a BYE request from the ‘A’-party, the ‘B’-party UA **SHALL** send 200OK final response and clear the OVR call.

g) if the OVR call is cleared from the 'B'-party, it **SHALL** send a BYE request to the 'A'-party, the 'A'-party UA **SHALL** respond with a 200OK final response and clear the OVR call.

If a two-party call is established because the 'B' party OVR call limit has been exceeded,

a) the 'B'-party **SHALL** send only the 'B'-party microphone audio (B mic) to the 'A'-party, but **SHALL** continue to send the Sum of 'B'-party's G/G plus A/G audio (B∑G/G + A/G) on the monitoring path to the calling parties of any other of its previously established incoming OVR calls (if they exist).

b) the 'B'-party **SHALL** sum the G/G audio plus microphone audio from the calling parties of any of its incoming OVR calls. This combined audio stream **SHALL** be sent on the forward path of its established outgoing OVR call towards the 'C'- party (should one exist).

Once the OVR call has been cleared by either the 'A'-party or the 'B'-party, ,

a) the 'B'-party **SHALL** issue a SIP INFO message (text/plain) containing its updated tail to its established outgoing OVR call with the 'C'-party (if it exists), in order to provide the 'C'-party with updated information about its current audio connections (see 3.1.7). The summed audio stream being sent on the forward path to the called party should reflect connections present in the tail terminating at the 'B'-party.

3.1.6 OVR Call Parameter Values (Timers)

Timer T0 **SHALL** operate at the 'A'-party UA during state IA-Awaiting-Confirmation. It **SHALL** be started on sending INVITE (Priority=" urgent", Subject="IA call") and stopped on receipt of 200 (OK) or 4xx-6xx final responses.

Timer T0 **SHALL** have a value of 2 seconds.

Note 8.
The timer T0 may be set to other values according to ANSP/FAA specific requirements.

3.1.7 OVR call Loop Closure Detection method

Any time a chain is updated (via INVITE or INFO) a 'B'-party shall compare the updated chain with the locally stored tail(s). Likewise any time a tail is updated (via INVITE or INFO) a 'B'-party shall compare the updated tail with the locally stored chain. In the case that there exists an intersection, (i.e. at least one identifier is an element of the chain and a tail), when comparing the chain with any tail terminating at 'B'-party, a loop closure has been detected.

Note 9.
In the case a loop has been detected via INFO, only a party that acted as 'B'-party in the last OVR call shall perform the following actions.

In the case the 'B'-party has detected a possible audio loop,

a) the 'B'-party **SHALL NOT** reject the INVITE request. The 'B'-party **SHALL** respond with a 100 Trying followed by a 200 OK response that contains an empty SDP "a=sid" attribute. Both the calling party and 'B'-party are then aware of a loop in the audio chain, but only B-party **SHALL** take action in order to prevent this from occurring.
b) the 'B'-party **SHALL** change its state to 'OVR-loop' and maintain a local intersection list containing all identifiers that are part of the updated tail and the chain.
c) the 'B'-party **SHALL NOT** issue a SIP INFO message (text/plain) containing its own updated tail to the established outgoing OVR call.
d) The 'B'-party **SHALL NOT** issue a SIP INFO message (text/plain) containing its own updated chain to the established incoming OVR call when a loop has been detected.

In the case that the 'B'-party has detected a loop in the audio chain, in order to avoid acoustic feedback caused by a closed audio loop, it **SHALL** perform the following actions:

- block all audio being received from calling party (that caused the loop);
- Send only the 'B'-party local Air-ground audio towards the calling party (i.e. B∑A/G)
- Send audio from any active DA/IDA telephone calls at the 'B'-party towards the calling user. `

The sent and/or received audio from all other incoming OVR calls to the 'B'-party that hasn't caused a loop **SHALL NOT** be affected.

3.1.8 OVR call Loop Closure Rejection method

Any time, a chain or tail is updated a 'B'-party set to 'OVR-loop' state shall compare the updated chain with the locally stored tail(s) or vice versa depending on which list had been received via INFO. In the case that there exists an updated intersection list and the set of elements comprising the new intersection (tail compared with chain) is less than the set of elements comprising the intersection when detecting the loop, (implying that the intersection is unequal to the previously derived intersection), then loop closure has been rejected.

In the case the 'B'-party has detected loop closure rejection,

a) the 'B'-party SHALL change to normal operation as described in 3.1.5.3.

3.1.9 OVR call A/G monitoring method

The system **SHALL** suppress the effects of oscillation, distortion, or level mis-regulation that might arise from OVR call chaining.

This could happen if an A/G audio stream is added to an existing OVR chain at two different operator positions. Within a system for Intercom (IC) OVR calls, this is not an issue, since a voice communication system has all information to handle this problem, but for Interphone (IP) OVR calls the problem can cause a distortion of A/G channels inserted several times into an OVR chain.

For example if the 'B'-party is monitoring frequencies f1 and f2, an established OVR call from the 'A'-party to the 'B'-party will result in the 'B'-party summing audio from the frequencies f1 and f2 plus any Ground-Ground audio etc. and forwarding it towards the 'A'-party. If the 'A'-party is also monitoring frequencies f1 and f2 and also has an established incoming OVR call, it would add in the same f1 and f2 audio already present within the summed stream, before sending it on. The same audio being added twice to the same stream at different points can cause distortion.

The following OVR call A/G monitoring method (reduced A/G monitoring channel) **SHALL** be followed:

a) Each called 'B'-party has information about all A/G channels in the OVR chain and **SHALL** add only A/G communications which are not in the A/G monitoring channel.

b) The called 'B'-party **SHALL** attenuate all A/G channels with a pre-configured value and send them together with the G/G channel to the calling 'A'-party.

c) In order to distribute the frequency information, the called 'B'-party **SHALL** add an SDP attribute "a=rid" in the 200 OK response that lists all Radio ID's that are currently included in the A/G monitoring channel. The elements of the list "a=rid:RID1 RID2 .. RIDn" **SHALL** be unique over space and time and listed as radio IDs (RIDx) separated by a single SPACE character.

Note 10.
It is expected that the Radio ID's associated with the RF frequencies will be defined by the ANSPs/FAA.

d) In case the local radio ID list needs to be updated (due to a new frequency being added or a previous frequency being removed), a SIP INFO message (text/plain) containing the updated radio ID list **SHALL** be sent to each established incoming OVR call.

e) On receiving a 200OK or a SIP INFO message containing the list of radio IDs (associated with frequencies) currently contained in the OVR A/G monitoring channel, a party **SHALL** add-in only new local A/G Radio IDs (associated with frequencies) that are not defined in its local radio ID list, before sending the updated list to each of its established incoming OVR calls.

f) A User Agent **SHALL** add audio from any of the local radio IDs (associated with frequencies) that are missing from the radio ID list to the audio in the OVR A/G monitoring channel.

Note 11.
It is foreseen that the above defined method has the advantage that the interphone OVR A/G monitoring channel uses only one RTP stream (with one audio channel) and therefore the required bandwidth is the same as for a normal DA/IDA interphone call. It also has the advantage because audio from each A/G frequency is only included once in an OVR A/G monitoring channel. Even in the case that audio from the same A/G frequency is also locally available, the attenuated A/G monitoring channel will have only reduced influence on the audio quality.

3.1.10 Interaction with Other ATS Supplementary Services

3.1.10.1 Interaction with the Call Priority Interruption Supplementary Service

An OVR call **SHALL NOT** be pre-empted.

3.1.10.2 Interaction with the Call Hold Supplementary Service

Call hold **SHALL NOT** be invoked for an OVR call.

3.1.10.3 Interaction with the Call Transfer Supplementary Service

Call transfer **SHALL NOT** be invoked for an OVR call.

3.1.10.4 Interaction with the Call Intrusion Supplementary Service

Intrusion **SHALL NOT** be invoked for an OVR call; this means that the Priority header field included in an INVITE for an OVR call **SHALL** always take value “urgent” and never “emergency”.

An OVR call **SHALL NOT** be intruded by a DA/IDA Priority call.

3.1.11 OVR call examples

3.1.11.1 OVR Call Chain and Tail operation example

Each node **SHALL** maintain a locally stored Chain and Tail list.
The tail list comprises all nodes forming a branch that terminate at the node in question. The chain list comprises all nodes forming a branch that originate from the node in question.

As new OVR calls are made or cleared at each SIP User Agent, the SIP User agent **SHALL** update its locally stored chain and tail lists to reflect the identities of the nodes in its tail(s) from which it is receiving summed audio on the forward path and also the identities of nodes in its chain from which it is receiving summed audio on the monitoring path.

On update of its locally stored tail list (due to establishment or clearing of OVR calls in the branch that terminates at the node), the node **SHALL** send an INFO message containing its updated locally stored tail list to the node to which it has established its outgoing OVR call (if one exists).

On update of its locally stored chain list (due to establishment or clearing of OVR calls in the branch that originates from the node), the node **SHALL** send an INFO message containing its updated locally stored chain list to all nodes with which it has established incoming OVR calls (if any exist).

The following figure provides an example of locally stored Chain and Tail list updates at nodes A,B,C and D when an OVR call is made from A to B, B to C and C to D. The 200OK response to an INFO message is not indicated in the diagram. The sequence of messages are indicated by numbers in brackets. The sequence of chain and tail list updates are indicated relative to message sequence numbers.

At any instant when the sequence of OVR calls are made, it can be observed that the locally stored chain list at any particular node defines the nodes forming the branch in front of the node, while the locally stored tail list at any particular node defines the nodes forming the branch behind the node.

- Send INVITE and INFO in forward direction with sid attribute defining Tail
- Send 200OK and INFO in backwards direction with sid attribute defining Chain

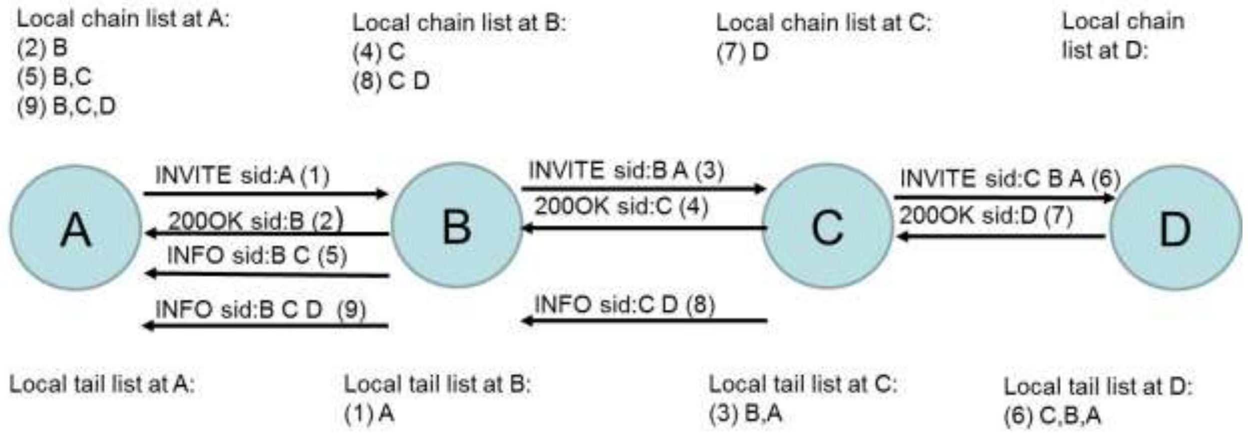


Figure 5 – Local chain and tail list update example for OVR calls - A to B, B to C & C to D.

Message sequence	Purpose
INVITE sid:A (1)	defines only local node A as there is no tail prior to A.
200OK sid:B (2)	defines only local node B as there is no chain after B (when OVR call from A arrives)
INVITE sid:B A (3)	defines local node B and the tail node A as defined in B's locally stored tail list
200OK sid:C (4)	defines only local node C as there is no chain after C (when OVR call from B arrives)
INFO sid:B C (5)	defines chain update sent from B to A as result of OVR call B→C
INVITE sid:C B A (6)	defines local node C and the tail nodes B and A as defined in C's locally stored tail list
200OK sid:D (7)	defines only local node D as there is no chain after D (when OVR call from C arrives)
INFO sid:C D (8)	defines chain update sent from C to B as result of OVR call C→D
INFO sid:B C D (9)	defines chain update sent from B to A as result of OVR call C→D

When making an OVR call the INVITE message sent by a node **SHALL** include an SDP a=sid: attribute that defines its own node identity plus all the node identities currently present in its locally stored tail list.

When answering an OVR call the 200OK response sent by a node **SHALL** include an SDP a=sid: attribute that defines its own node identity plus all the node identities currently present in its locally stored chain list.

For example an 'A'-party that has no established incoming OVR call connections **SHALL** have an SDP message body that includes the a=sid: attribute defining only its own node identity:

a=sid:IDA

while the 200OK response sent by the 'B'-party that also has no outgoing OVR call connections **SHALL** have an SDP message body that includes the a=sid: attribute defining the node identity of the 'B'-party.

a=sid:IDB

On establishment of the above OVR call, the 'A'-party will have a locally stored chain list defining IDB, while the 'B'-party will have a locally stored tail list defining IDA.

As another example, an 'A'-party that already has other direct or indirect incoming OVR call connections to a 'C'-party and a 'D'-party **SHALL** have an SDP message body that includes the a=sid: attribute defining its own node identity plus its tail node identities. Each node identity is separated by one space (without any comma) as below:

a=sid:IDA IDC IDD

while the 200OK response sent by the 'B'-party that also has established an outgoing OVR call to a 'E'-party which in turn has established its outgoing OVR call to an 'F'-party **SHALL** have an SDP message body that includes the a=sid: attribute defining its own identity plus its chain node identities. Each node identity is separated by one space (without any comma) as below:

a=sid:IDB IDE IDF

On establishment of the above OVR call, the 'A'-party will now have a locally stored chain list defining IDB, IDE and IDF, while the 'B'-party will now have a locally stored tail list defining IDA, IDC and IDD.

3.1.11.2 OVR Call loop detection example

The following figure provides an example of audio loop detection. The sequence of events leading up to an audio loop detection in the example are defined as follows:

1. “B” makes OVR call to “C”
2. “C” makes OVR call to “D”
3. “D” makes OVR call to “A”
4. “A” makes OVR call to “B”
5. “B” detects loop as Tail list from “A” matches at least one element defined in its current Chain list.
6. “B” creates Local intersection list: Tail (A,D,C.B) ∩ Chain list (C,D,A) = C,D,A
7. “B” sends empty a=sid: attribute to “A” and blocks audio received on forward and receive paths from being summed, sending only the Sum of “B”s A/G audio towards “A”.
8. On receiving an empty a=sid: attribute without any node identities, “A” does not update its local Chain list and therefore doesn’t send INFO message with updated chain,
9. “B” does not send INFO message to “C” with updated tail

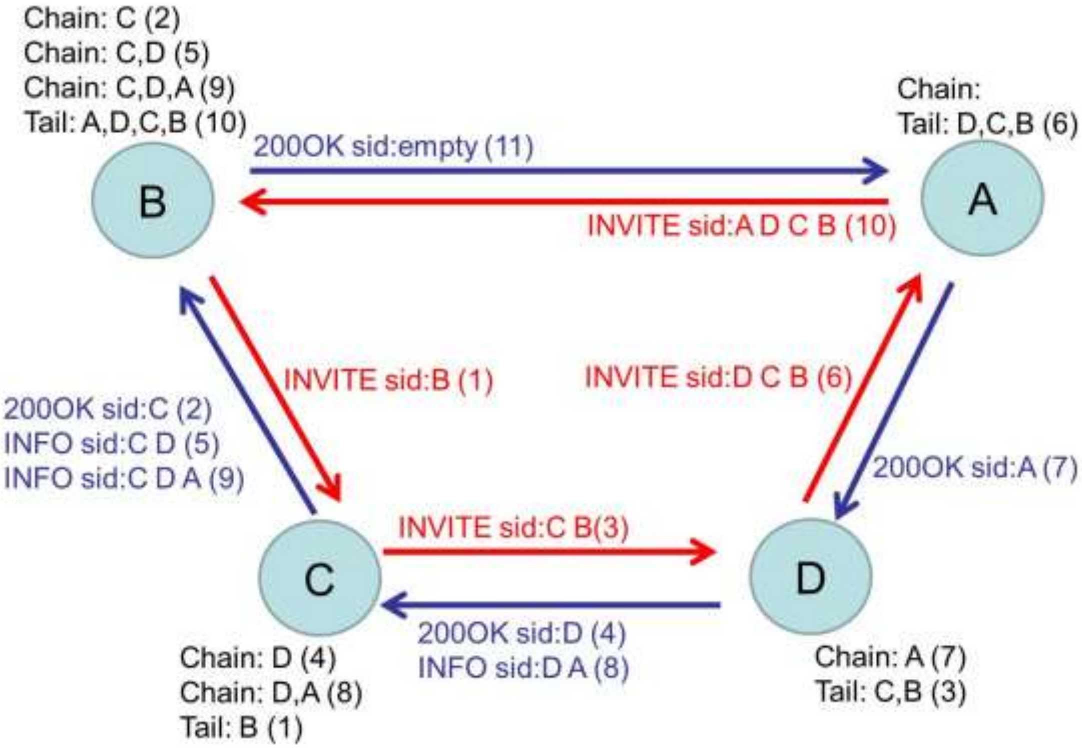


Figure 6 – Audio Loop detection by the ‘B’-node

The following figure relates to the previous example and shows the summed audio flows and blocks imposed by B following audio loop detection:

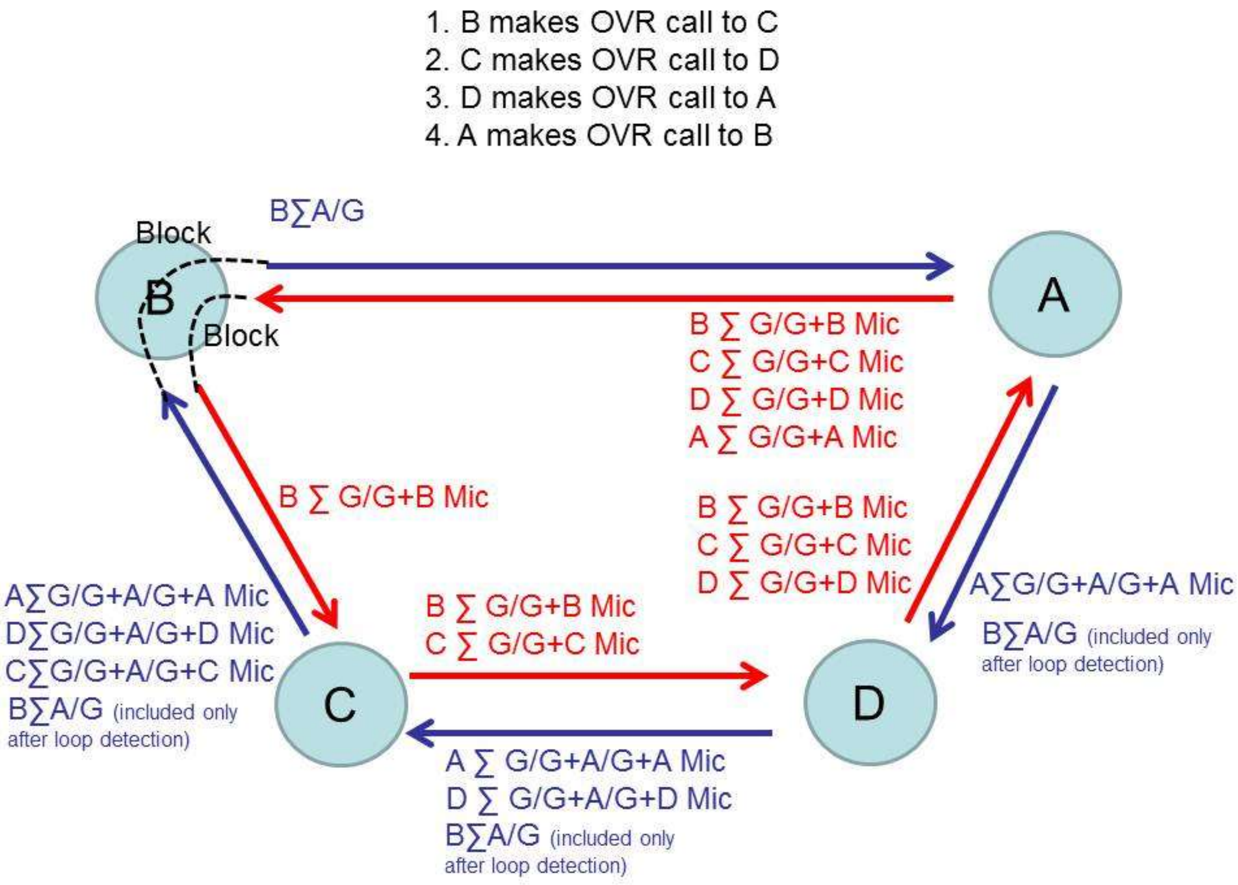


Figure 7 –Audio loop detected by B that blocks transit of forward & monitoring path audio

Following the audio block imposed by B when an audio loop has been detected, it can be observed from the above figure each of the individual parties can hear the following audio:

A hears:(A∑G/G+A/G+Amic)+(B∑G/G+A/G+Bmic)+C∑G/G+Cmic)+(D∑G/G+Dmic)
D hears:(A∑G/G+A/G+Amic)+(B∑G/G+A/G+Bmic)+C∑G/G+Cmic)+(D∑G/G+A/G+Dmic)
C hears:(A∑G/G+A/G+Amic)+(B∑G/G+A/G+Bmic)+C∑G/G+A/G+Cmic)+(D∑G/G+A/G+Dmic)
B hears:(A∑G/G+A/G+Amic)+(B∑G/G+A/G+Bmic)+C∑G/G+A/G+Cmic)+(D∑G/G+A/G+Dmic)

3.1.11.3 OVR Call loop rejection example

The following figure progresses from the previous example to show the audio loop rejection when the audio loop is opened as a result of an OVR call forming part of the loop being cleared. The sequence of events leading up to an audio loop rejection as defined as follows:

1. A clears OVR call with D, this will trigger the following:
2. D updates its locally stored Chain list and sends an INFO defining the updated chain to C;
3. C updates its locally stored Chain list and sends an INFO defining the updated chain to B;
4. A updates its locally stored Tail list and sends an INFO defining the updated tail to B;
5. B updates its locally stored Tail list;
6. If tail update arrives at B first, B compares new tail (A) with Chain (C,D,A) and rejects loop because new intersection is unequal to the previously derived intersection;
7. If chain update arrives at B first, B compares new chain (C D) with Tail (A,D,C,B) and rejects loop because new intersection is unequal to the previously derived intersection;
8. On loop rejection B unblocks audio received on its forward and receive paths from being summed with locally received audio;
9. B is now enabled to send an INFO defining the updated tail to C and an INFO defining the updated chain to A;
10. C updates its locally stored Tail list and sends an INFO defining the updated tail to D;

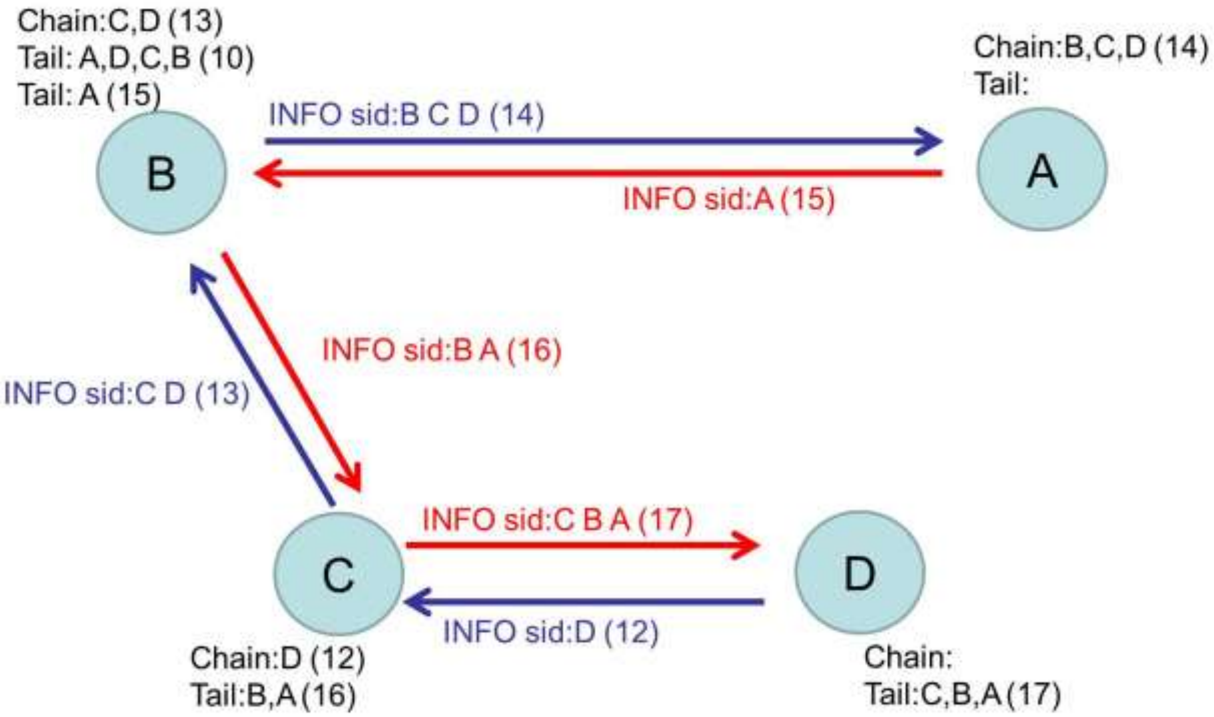


Figure 8 –Audio Loop rejection by 'B'-node

3.1.11.4 OVR Call ‘A’ to ‘B’ and ‘B’ to ‘A’ example

In the following scenario, the ‘A’-party firstly establishes an OVR call to the ‘B’-party.
The ‘A’-party will have a locally stored chain list defined as ‘B’ while the ‘B’-party will have a locally stored Tail list defined as ‘A’.
The ‘B’-party then establishes an OVR call to the ‘A’-party. As an Override call request can’t be refused, the OVR call to the ‘A’-party will proceed. When however the ‘A’-party receives the OVR call request from the ‘B’-party, it will apply an audio loop detection check to determine if the OVR call establishment would cause an audio loop to occur. To do this it compares the Tail element it receives in the INVITE from the ‘B’-party with the locally stored chain list defined at the ‘A’-party and verifies if there are common elements. In this case its intersection list will therefore define one element ‘B’. An audio loop has therefore been detected and the ‘B’-party will enter in the ‘OVR-loop’ state

In order to prevent acoustic feedback, the ‘A’-party **SHALL** block audio received on the forward path from the ‘B’-party (B∑G/G +Bmic) from being summed with audio sent on the forward path to the ‘B’-party. The ‘A’-party SHALL also block the monitored audio received on the path from the ‘B’-party (B∑G/G+A/G +Bmic) from being summed with audio on the monitored path towards the ‘B’-party, just sending the ‘A’-party Air-ground audio (A∑A/G). The result is that the ‘A’-party hears B ∑ G/G+A/G+B Mic, while the ‘B’-party hears A ∑ G/G+A/G+A Mic.

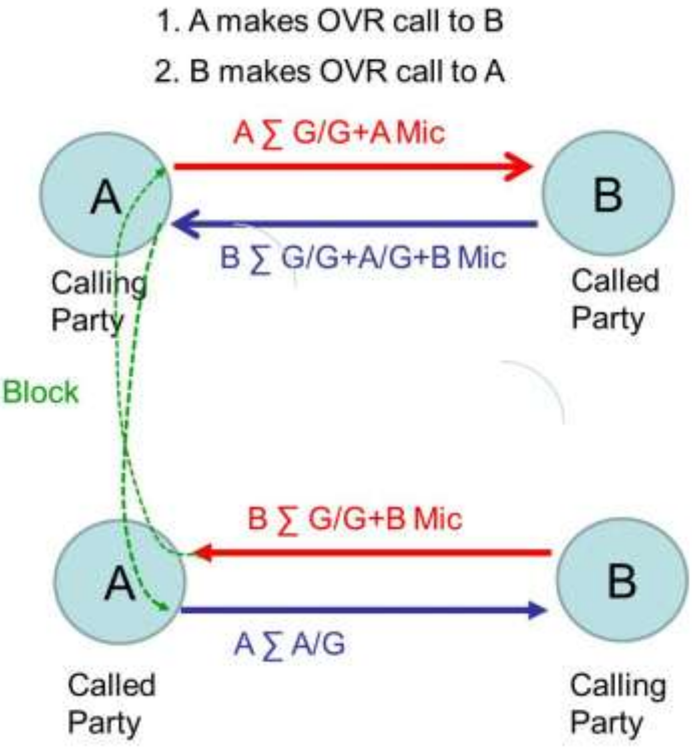


Figure 9 –‘A’ calls ‘B’ and ‘B’ calls ‘A’ loop detection example

3.1.11.5 OVR Call Radio ID example

The example below shows that the ‘C’-party is monitoring frequencies f1 and f2, the ‘B’-party is monitoring f1,f2 and f4, while the ‘A’-party is monitoring just f5. An OVR call from the ‘B’-party to the ‘C’-party results in the ‘C’-party sending the summed audio of f1 and f2 on monitoring channel towards the ‘B’-party. An OVR call from the ‘A’-party to the ‘B’-party, implies that the ‘B’-party would add its own frequencies to the sum sent from the ‘C’-party, before sending the summed audio on the monitoring channel to the ‘A’-party. The ‘B’-party realizes however that the summed audio it is receiving from the ‘C’-party already contains f1 and f2 frequencies. The ‘B’-party therefore just adds the frequency f4 to the sum, that it sends on the monitoring channel towards the ‘A’-party.
At the same time, B-party will continue listening to frequencies f1, f2 and f4.

Later on the ‘C’-party adds a new frequency f3 and sends an INFO message to inform the ‘B’-party of the summed frequencies being sent from the ‘C’-party. The ‘B’-party in turn will continue to add only the frequency f4 to the sum and sends an INFO message to inform the ‘A’-party of the summed frequencies being sent from the ‘B’-party.

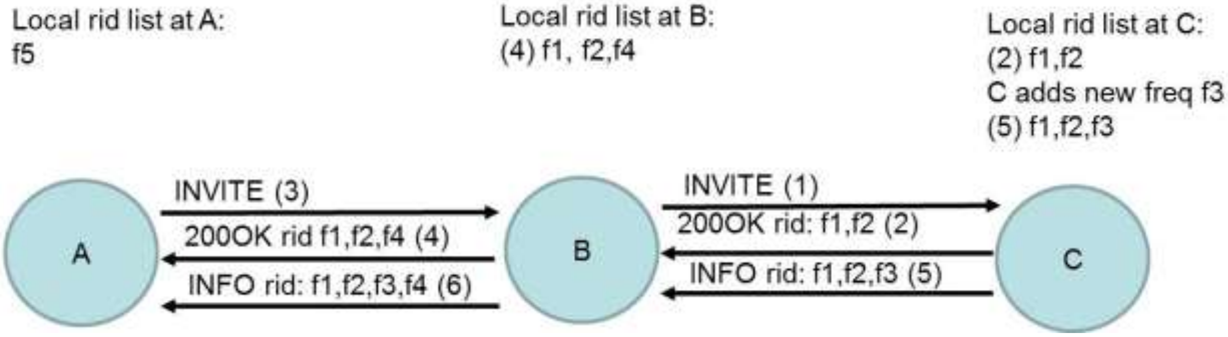


Figure 10 –Local rid list example for OVR calls- B to C, A to B

3.1.12 OVR Call Message Sequence Charts

The Message Sequence Charts (MSC) in figures below show the information flows between the 'A'-party UA and the 'B'-party UA. Each information flow is named according to the corresponding message sent to or received from respective party.

Dashed lines (---) represent SIP signaling messages that are mandatory to the call scenario. The arrow indicates the direction of message flow.

Double dashed lines (===) represent media paths between network elements.

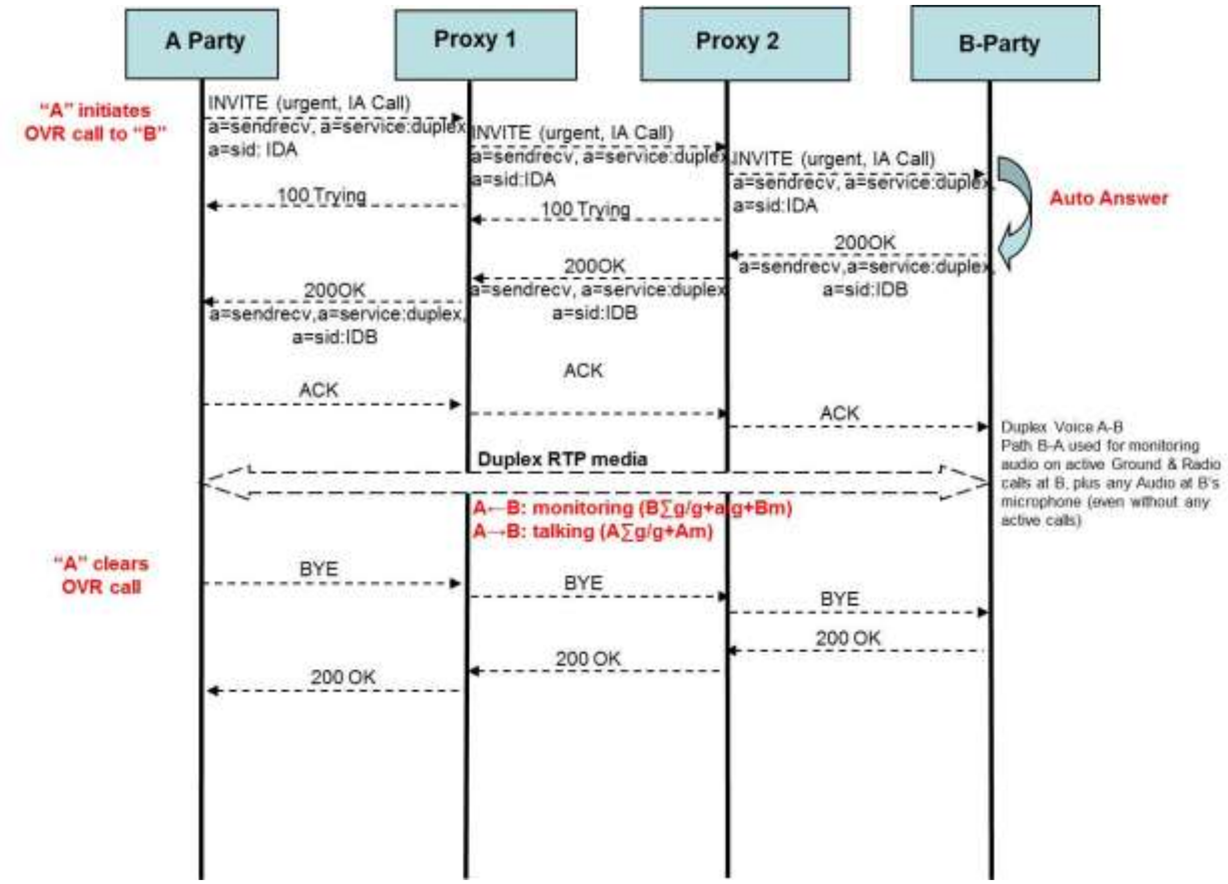

A Party
Proxy 1
Proxy 2
B-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid: IDA
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDA
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDA
Auto Answer
100 Trying
100 Trying
200OK
a=sendrecv,a=service:duplex
a=sid:IDB
200OK
a=sendrecv, a=service:duplex
a=sid:IDB
200OK
a=sendrecv,a=service:duplex,
a=sid:IDB
ACK
ACK
ACK
Duplex RTP media
Duplex Voice A-B
Path B-A used for monitoring audio on active Ground & Radio calls at B, plus any Audio at B's microphone (even without any active calls)
A←B: monitoring (B∑g/g+a/g+Bm)
A→B: talking (A∑g/g+Am)
"A" clears OVR call
BYE
BYE
BYE
200 OK
200 OK
200 OK

Figure 11 - OVR call from A-party to B-Party, OVR call established and A-Party clears call

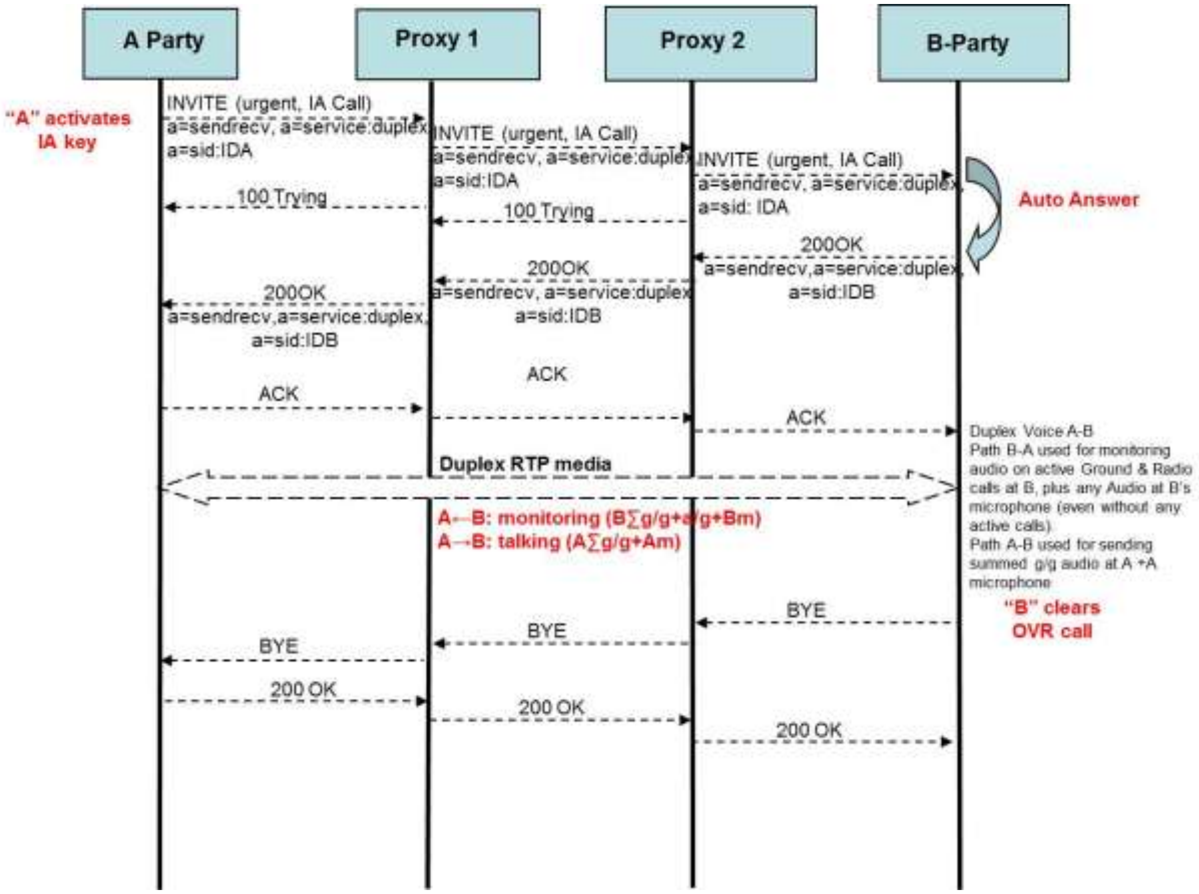

A Party
Proxy 1
Proxy 2
B-Party
"A" activates IA key
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDA
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDA
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex,
a=sid: IDA
Auto Answer
100 Trying
100 Trying
200OK
a=sendrecv,a=service:duplex,
a=sid:IDB
200OK
a=sendrecv, a=service:duplex
a=sid:IDB
200OK
a=sendrecv,a=service:duplex,
a=sid:IDB
ACK
ACK
ACK
Duplex RTP media
A←B: monitoring (B∑g/g+a/g+Bm)
A→B: talking (A∑g/g+Am)
Duplex Voice A-B
Path B-A used for monitoring audio on active Ground & Radio calls at B, plus any Audio at B's microphone (even without any active calls)
Path A-B used for sending summed g/g audio at A +A microphone
"B" clears OVR call
BYE
BYE
BYE
200 OK
200 OK
200 OK

Figure 12 - OVR call from A-party to B-Party, OVR call established and B-Party clears call

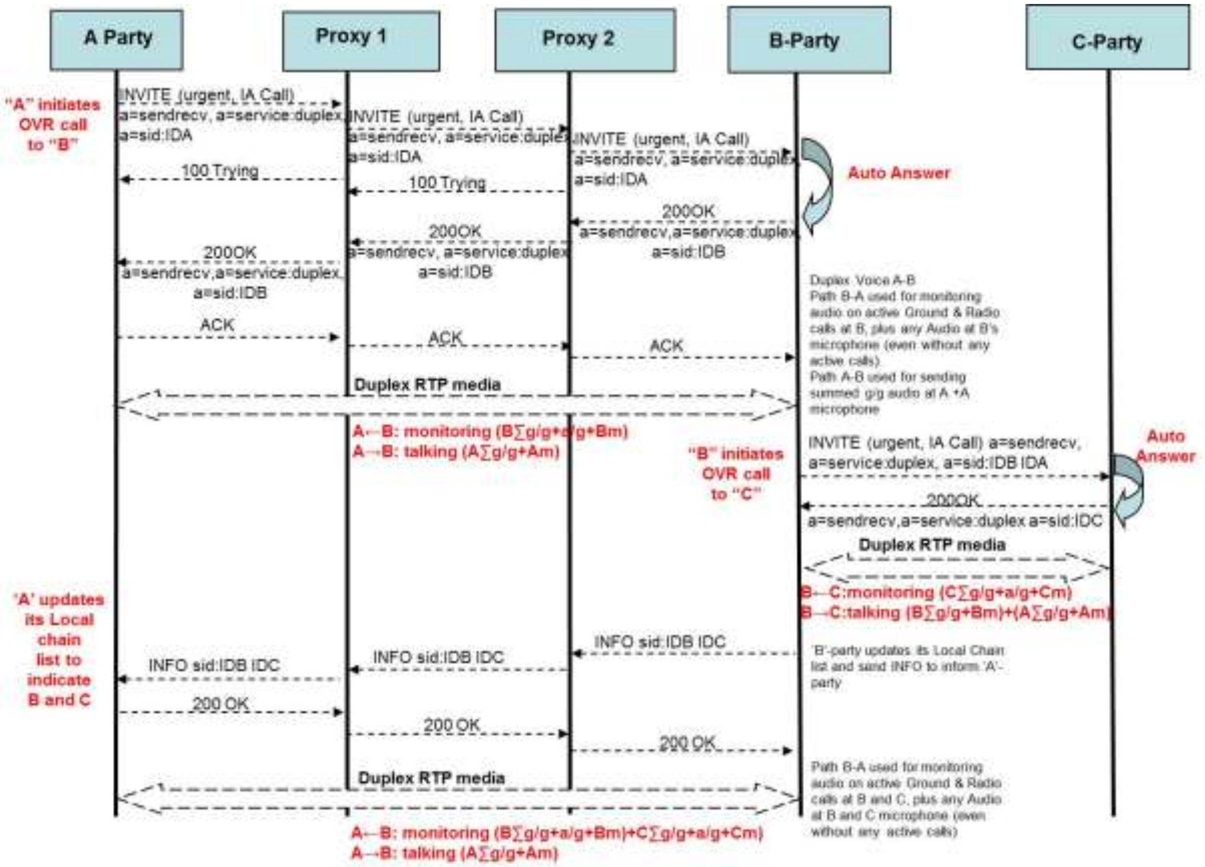

A Party
Proxy 1
Proxy 2
B-Party
C-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call) a=sendrecv, a=service:duplex a=sid:IDA
100 Trying
200OK a=sendrecv,a=service:duplex a=sid:IDB
ACK
Auto Answer
Duplex Voice A-B
Path B-A used for monitoring audio on active Ground & Radio calls at B, plus any Audio at B's microphone (even without any active calls)
Path A-B used for sending summed g/g audio at A +A microphone
Duplex RTP media
A←B: monitoring (B∑g/g+a/g+Bm)
A→B: talking (A∑g/g+Am)
"B" initiates OVR call to "C"
INVITE (urgent, IA Call) a=sendrecv, a=service:duplex, a=sid:IDB IDA
Auto Answer
200OK a=sendrecv,a=service:duplex a=sid:IDC
B←C:monitoring (C∑g/g+a/g+Cm)
B→C:talking (B∑g/g+Bm)+(A∑g/g+Am)
'A' updates its Local chain list to indicate B and C
INFO sid:IDB IDC
'B'-party updates its Local Chain list and send INFO to inform 'A'-party
200 OK
Path B-A used for monitoring audio on active Ground & Radio calls at B and C, plus any Audio at B and C microphone (even without any active calls)
A←B: monitoring (B∑g/g+a/g+Bm)+C∑g/g+a/g+Cm)
A→B: talking (A∑g/g+Am)

Figure 13 - OVR call from A-party to B-Party,
B-Party then establishes outgoing OVR call to C-Party

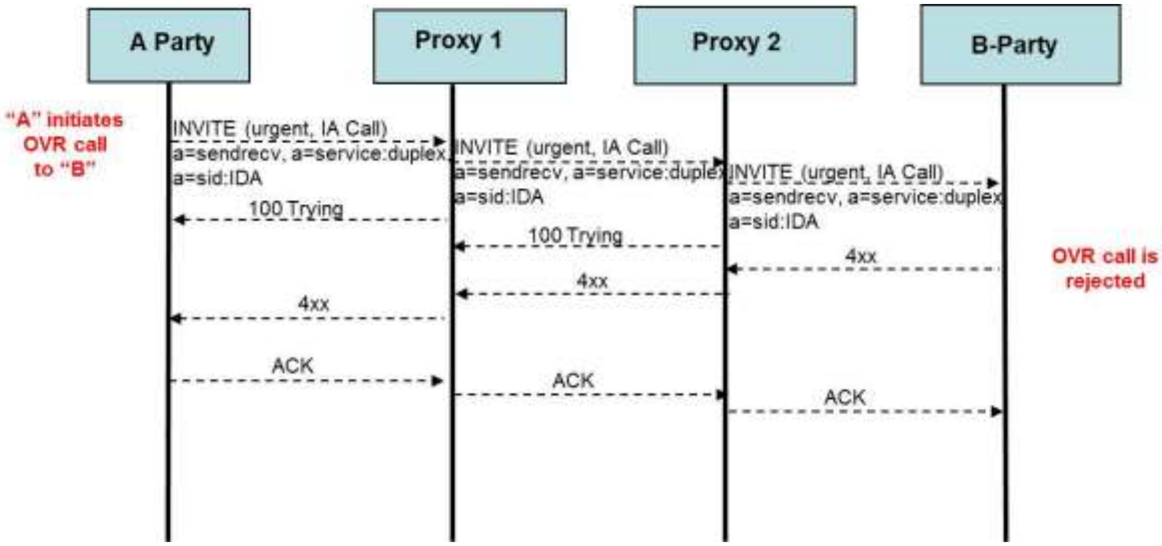

A Party
Proxy 1
Proxy 2
B-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call) a=sendrecv, a=service:duplex a=sid:IDA
100 Trying
4xx
ACK
OVR call is rejected

Figure 14 - OVR call from A-party to B-Party,
OVR call is rejected (i.e. unrecognized address)

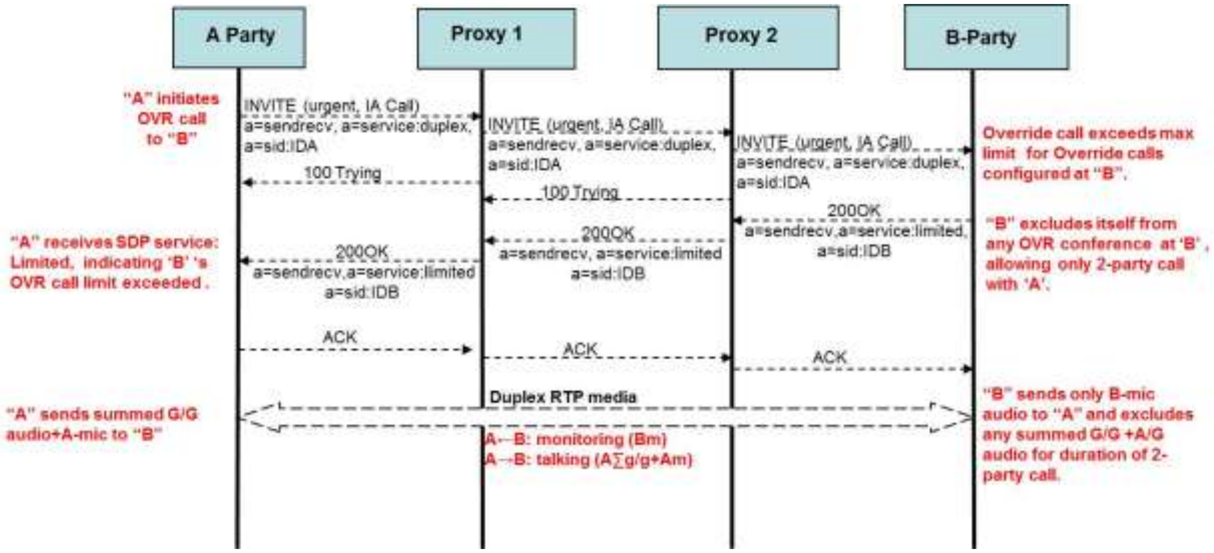

A Party
Proxy 1
Proxy 2
B-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex,
a=sid:IDA
100 Trying
200OK
a=sendrecv,a=service:limited
a=sid:IDB
ACK
Duplex RTP media
A←B: monitoring (Bm)
A→B: talking (A∑g/g+Am)
Override call exceeds max limit for Override calls configured at "B".
"B" excludes itself from any OVR conference at 'B', allowing only 2-party call with 'A'.
"A" receives SDP service: Limited, indicating 'B' 's OVR call limit exceeded.
"A" sends summed G/G audio+A-mic to "B"
"B" sends only B-mic audio to "A" and excludes any summed G/G +A/G audio for duration of 2-party call.

Figure 15 - OVR call from A-party to B-Party that exceeds B's max OVR call limit , results in a 2-party call for call duration

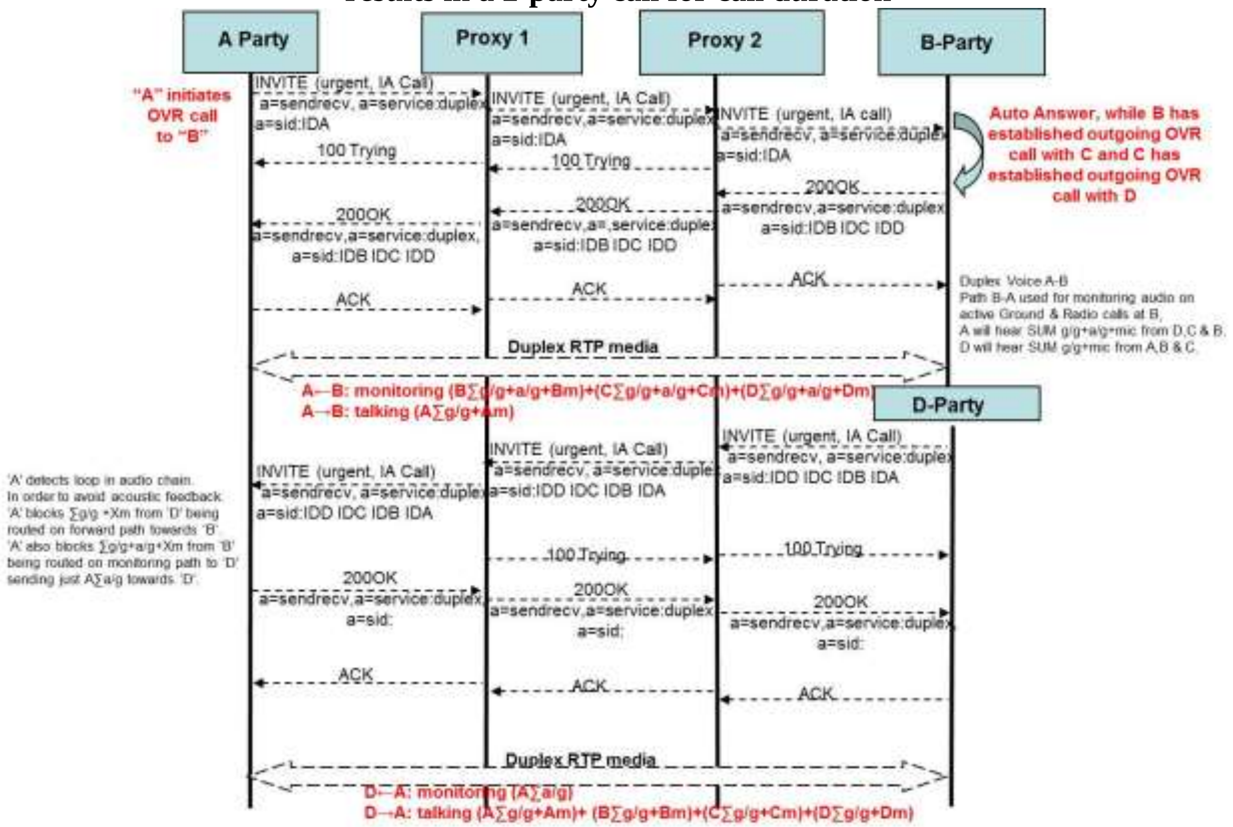

A Party
Proxy 1
Proxy 2
B-Party
D-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDA
100 Trying
200OK
a=sendrecv,a=service:duplex,
a=sid:IDB IDC IDD
ACK
Auto Answer, while B has established outgoing OVR call with C and C has established outgoing OVR call with D
Duplex Voice A-B
Path B-A used for monitoring audio on active Ground & Radio calls at B,
A will hear SUM g/g+a/g+mic from D,C & B,
D will hear SUM g/g+mic from A,B & C
Duplex RTP media
A←B: monitoring (B∑g/g+a/g+Bm)+(C∑g/g+a/g+Cm)+(D∑g/g+a/g+Dm)
A→B: talking (A∑g/g+Am)
'A' detects loop in audio chain.
In order to avoid acoustic feedback
'A' blocks ∑g/g +Xm from 'D' being routed on forward path towards 'B'
'A' also blocks ∑g/g+a/g+Xm from 'B' being routed on monitoring path to 'D'
sending just A∑a/g towards 'D'.
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
a=sid:IDD IDC IDB IDA
100 Trying
200OK
a=sendrecv,a=service:duplex
a=sid:
ACK
Duplex RTP media
D←A: monitoring (A∑a/g)
D→A: talking (A∑g/g+Am)+ (B∑g/g+Bm)+(C∑g/g+Cm)+(D∑g/g+Dm)

Figure 16 - OVR call from A to B while B has outgoing OVR call established with C, and C with D. OVR call from D to A will cause loop in audio chain

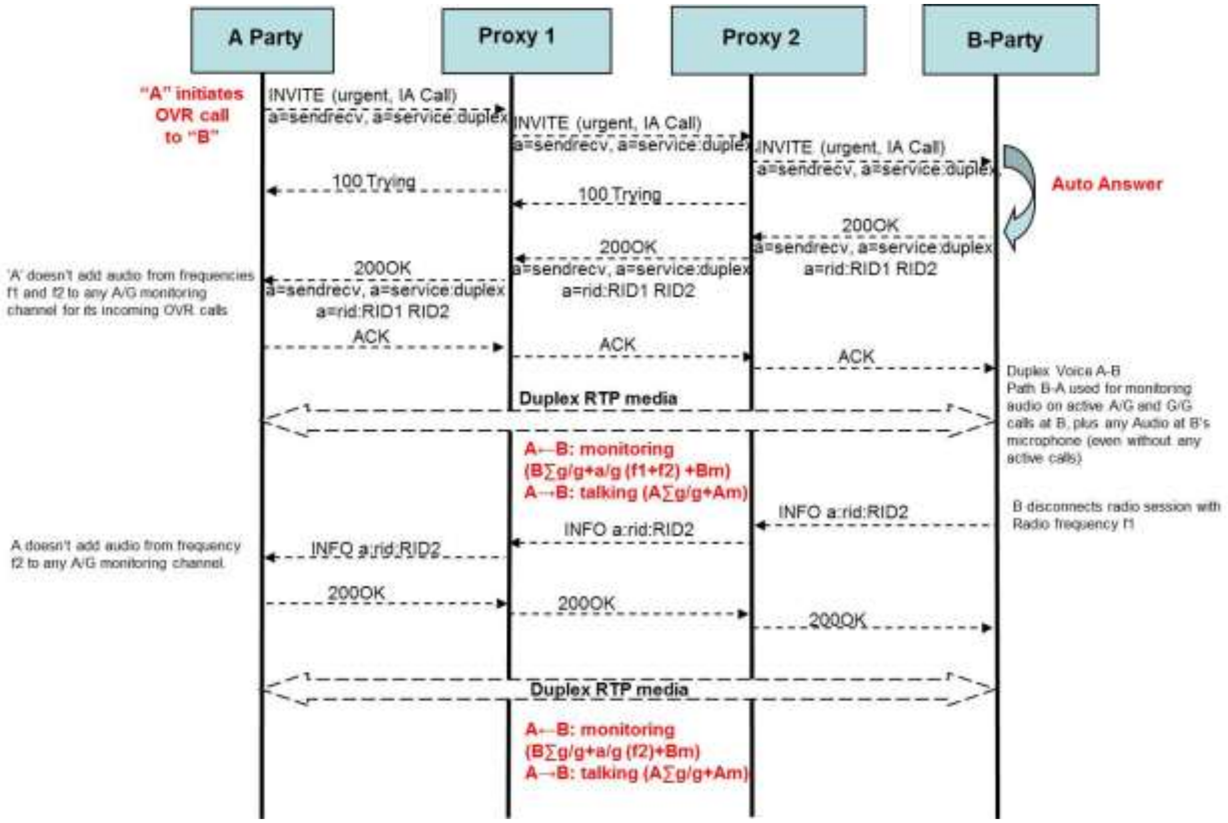
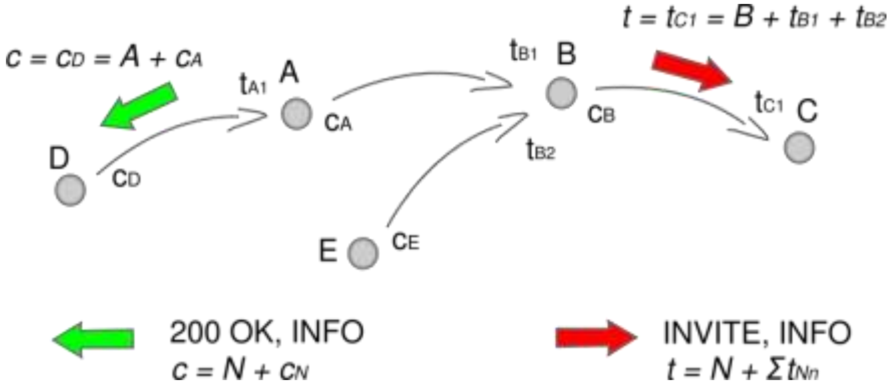

A Party
Proxy 1
Proxy 2
B-Party
"A" initiates OVR call to "B"
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
INVITE (urgent, IA Call)
a=sendrecv, a=service:duplex
Auto Answer
100 Trying
100 Trying
200OK
a=sendrecv, a=service:duplex
a=rid:RID1 RID2
200OK
a=sendrecv, a=service:duplex
a=rid:RID1 RID2
'A' doesn't add audio from frequencies f1 and f2 to any A/G monitoring channel for its incoming OVR calls
200OK
a=sendrecv, a=service:duplex
a=rid:RID1 RID2
ACK
ACK
ACK
Duplex Voice A-B
Path B-A used for monitoring audio on active A/G and G/G calls at B, plus any Audio at B's microphone (even without any active calls)
Duplex RTP media
A←B: monitoring
(B∑g/g+a/g (f1+f2) +Bm)
A→B: talking (A∑g/g+Am)
B disconnects radio session with Radio frequency f1
INFO a:rid:RID2
INFO a:rid:RID2
A doesn't add audio from frequency f2 to any A/G monitoring channel.
INFO a:rid:RID2
200OK
200OK
200OK
Duplex RTP media
A←B: monitoring
(B∑g/g+a/g (f2)+Bm)
A→B: talking (A∑g/g+Am)

Figure 17-OVR call from A party to B party while B has radio sessions with f1 and f2

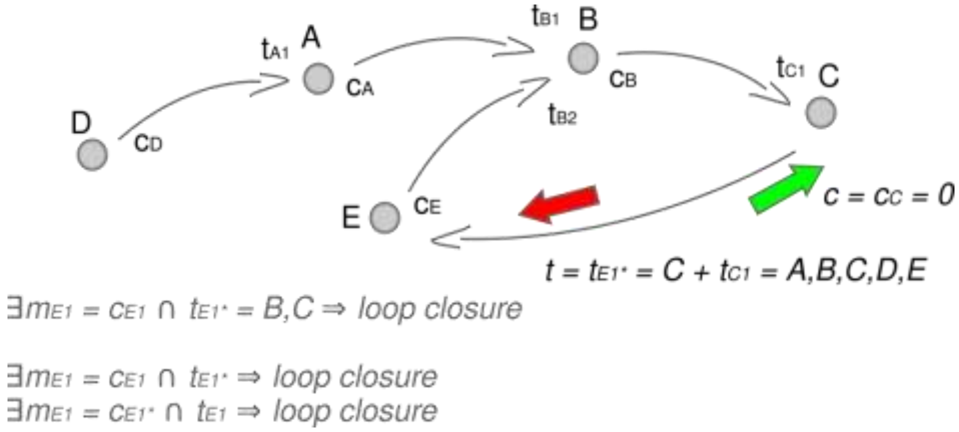
ANNEX A: OVERRIDE CALL CHAIN AND TAIL THEORY

Each node shall provide a tail and a chain list to its adjacent node. The tail comprises all nodes that form a branch terminating at the overridden node, e.g. [*D*, *A*, *E* are tails of *B]*. The chain comprises all nodes that form a branch originating from an overriding node, e.g. [*B*, *C* is the chain of *A]*, see the figure below.

In general, a tail is being forwarded via INVITE and INFO messages containing the sending node identifier (*N*) and the sum of all tails connecting to this node ($t = N + \sum t_{Nn}$) as a list within the SDP attribute a=sid:N, N_1, N_2,…, N_n. The chain is being forwarded via 200 OK (as a final response to the INVITE request) and INFO messages containing the sending node identifier (*N*) and the chain connecting to this node ($c = N + c_N$).



Any time, a chain or tail is updated a node shall compare the updated chain with the locally stored tail(s) or vice versa depending on which list had been received. In the case that there exists an intersection m_{Nn} when comparing the chain (c_N) with any tail (t_{Nn}) of the node (*N*), meaning that there is at least one node identifier which is an element of the chain and the tail (t_n), a loop closure has been detected for the tail (t_n).

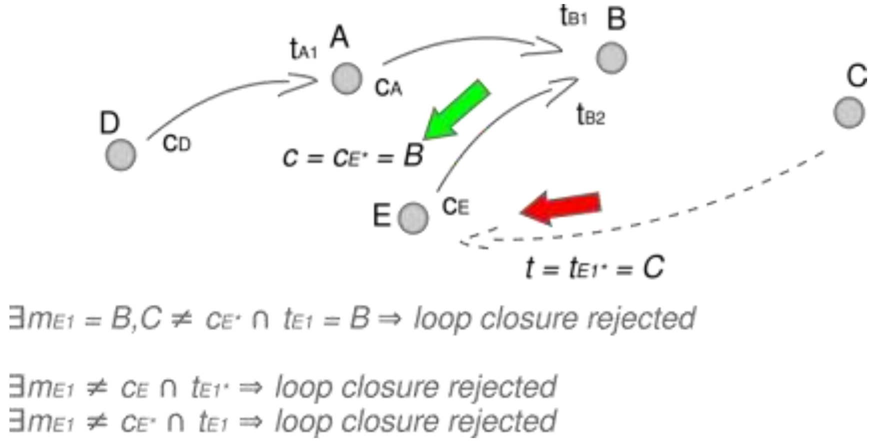

tA1
A
cA
tB1
B
cB
tB2
tC1
C
D
cD
E
cE
c = cC = 0
t = tE1* = C + tC1 = A,B,C,D,E
∃mE1 = cE1 ∩ tE1* = B,C ⇒ loop closure
∃mE1 = cE1 ∩ tE1* ⇒ loop closure
∃mE1 = cE1* ∩ tE1 ⇒ loop closure

For instance node *E* would detect a loop closure as soon as node *C* opens an OVR call towards node *E*, see the previous figure.

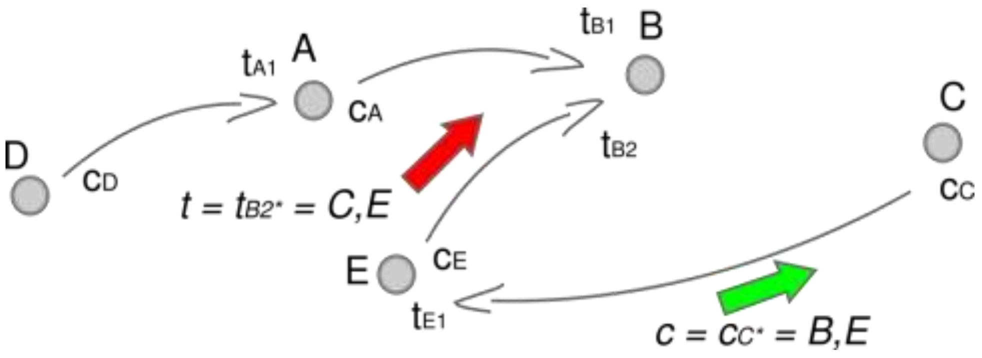
In this case, node *E* shall change its state to “loop closure” meaning that it responds to *C* with an empty chain (a=sid) as part of the 200 OK response and changes its audio routing accordingly. In addition, *E* does not forward the updated tail (t_{E1*}, received from *C*) to *B*. In the case, *C* receives

an empty chain (a=sid), *C* does (per definition) not update its local chain but still forwards tail updates to *E that have been received from other neighbors*.
Considering an established OVR call, e.g. between A and B, an incoming INFO message received at B that requires updating the tail at B may lead to a loop detection triggering the same behavior at B as described above.

In the case the loop is opened by other nodes, e.g. between *B* and *C*, *E* receives a chain update from *B* and a tail update from *C*. Node E shall again compare chain and tail or vice versa depending on which update is received first.
In the case m_{E1} is unequal to the intersection as a result comparing tail and chain loop closure is rejected. In this example [*B*, *C*] is unequal to [*B*] (assuming that the tail has been updated first). *E* changes back to "normal state" meaning that m_{E1} *is deleted,* audio is routed accordingly and list updates (tail and chain) are forwarded to adjacent nodes.


tA1
A
cA
tB1
B
tB2
C
D
cD
c = cE* = B
E
cE
t = tE1* = C
∃mE1 = B,C ≠ cE* ∩ tE1 = B ⇒ loop closure rejected
∃mE1 ≠ cE ∩ tE1* ⇒ loop closure rejected
∃mE1 ≠ cE* ∩ tE1 ⇒ loop closure rejected

Node *E* forwards the updated chain to node *C* via INFO and thus *C* updates the local list and forwards accordingly. The updated tail is sent from *E* via INFO to node *B*, see figure below.


tA1
A
cA
tB1
B
tB2
C
cC
D
cD
t = tB2* = C,E
E
cE
tE1
c = cC* = B,E